Multi-Model Algorithms for Optimization

Richard Gordon Carter

May 1986

TR86-3

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE
MAY 1986

2. REPORT TYPE

3. DATES COVERED
00-00-1986 to 00-00-1986

4. TITLE AND SUBTITLE
Multi-Model Algorithms for Optimization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computational and Applied Mathematics Department ,Rice University,6100 Main Street MS 134,Houston,TX,77005-1892

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**182**	

RICE UNIVERSITY

MULTI-MODEL ALGORITHMS FOR OPTIMIZATION

by

RICHARD GORDON CARTER

A Thesis Submitted
In Partial Fulfillment Of The
Requirements For The Degree

DOCTOR OF PHILOSOPHY

Approved, Thesis Committee:

JE Dennis Jr.

John E. Dennis Jr.,
Professor of Mathematical Sciences
Chairman

Richard A Tapia

Richard A. Tapia,
Professor of Mathematical Sciences

Richard D Young

Richard D. Young,
Professor of Economics

Houston, Texas

May, 1986

ACKNOWLEDGEMENTS

This thesis was only possible through the support of colleagues and friends. Thanks go to my committee: John Dennis, Richard Tapia and Richard Young. In particular, both ears and the tail should be awarded to John Dennis for his aid in clarifying the text presented herein.

For introducing me to the field of Numerical Analysis, I thank my good friend Jim Solomon. For aid in the preparation of the manuscript, I am indebted to Vivian Choi and Luis Beguiristain. Special praise goes to Bill LeFebvre for his tireless efforts with the laser printer.

This research was supported by NSF grant MCS-81-16779 and by AFOSR85-0243.

Revised edition, 6/3/86

MULTI-MODEL ALGORITHMS
FOR
OPTIMIZATION

by

RICHARD GORDON CARTER

ABSTRACT

A recent approach for the construction of nonlinear optimization software has been to allow an algorithm to choose between two possible models to the objective function at each iteration. The model switching algorithm NL2SOL of Dennis, Gay and Welsch and the hybrid algorithms of Al-Baali and Fletcher have proven highly effective in practice. Although not explicitly formulated as multi-model methods, many other algorithms implicitly perform a model switch under certain circumstances (e.g., resetting a secant model to the exact value of the Hessian).

We present a trust region formulation for multi-model methods which allows the efficient incorporation of an arbitrary number of models. Global convergence can be shown for three classes of algorithms under different assumptions on the models. First, essentially any multi-model algorithm is globally convergent if each of the models is sufficiently well behaved. Second, algorithms based on the central feature

of the NL2SOL switching system are globally convergent if one model is well behaved and each other model obeys a ``sufficient predicted decrease'' condition. No requirement is made that these alternate models be quadratic. Third, algorithms of the second type which directly enforce the ``sufficient predicted decrease'' condition are globally convergent if a single model is sufficiently well behaved.

CONTENTS

CHAPTER 1

General Introduction : Multi-model Algorithms for Optimization

1.1. Introduction

In this thesis, we are concerned with a class of algorithms for the solution of problems of the form:

$$\text{minimize } f(x) \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^1$ is twice continuously differentiable with gradient $g(x)$ and Hessian $H(x)$. We refer to this as the **unconstrained minimization problem.** Historically, many different solution algorithms have been created for this problem, most of which are based on different ways of creating a local model of the objective function. At each iteration, some scheme produces a single model for the objective function and uses this to determine the next approximation to the function minimizer. Consider Newton's method for solving (1.1).

Algorithm A.1.1 : Newton's Method

```
0) Initialize.
1) Repeat until convergence :
    Set $x_{k+1} = x_k - H(x_k)^{-1} g(x_k)$ .
2) End.
```

For a convex[1] function f , this is equivalent to the following algorithm.

Algorithm A.1.2 : Newton's Method

```
0) Initialize.
1) Repeat until convergence [2] :
    Set $x_{k+1} = \underset{x}{argmin}\{ f(x_k) + g(x_k)^t (x - x_k) + \frac{1}{2}(x - x_k)^t H(x_k)(x - x_k) \}$.
2) End .
```

The function f is modeled at each iteration by the Newton model, which is the first three terms of a Taylor expansion. New iterates are generated by computing the minimizer of this model.

Since most minimization algorithms have been based on quadratic local models, the process of "producing a model" has essentially been that of determining an approximation to the Hessian of the objective function. If the exact Hessian was

[1] A function is said to be convex if for every pair of vectors $x, y \in \mathbb{R}^n$ and every $\xi \in [0,1]$ it follows that $f(\xi x + (1-\xi)y) \leq \xi f(x) + (1-\xi) f(y)$.

[2] The notation x^t denotes the transpose of the vector x. The notation $\underset{x}{argmin}\{ h(x) \}$ denotes the value of x which minimizes the function $h(x)$ provided such a minimizer exists and is unique.

available, the Newton model could be used. If it was not, or if it was too expensive to calculate, then a finite difference Hessian approximation could be used. If this seemed to be too expensive, one could turn to one of the secant methods, such as the well known BFGS and DFP updates.[3] If the problem had special structure, other models could be used to take advantage of this.

An alternative approach has begun appearing in the literature over the last few years, notably in the algorithm NL2SOL of Dennis, Gay, Welsch [1981 a,b] and the works of Nazareth [1980, 1983], Al-Baali, Fletcher [1985], and Dennis, Sheng, Vu [1985]. In this approach, an algorithm considers more than one possible model at a given stage of the iteration. Nazareth, Fletcher, and Al-Baali call such algorithms *hybrid methods,* but we refer to them as *multi-model* algorithms. In simplest form, algorithms of this type are structured as follows.

Algorithm A.1.3 : A Multi-Model Algorithm

```
0) Initialize.
1) Repeat until convergence :
     2) Choose a model from two or more possibilities.
     3) Compute an approximate minimizer for this model.
     4) Either accept this minimizer as $x_{k+1}$ and proceed, or
                 return to 2) and try another model.
5) End .
```

[3] See, for example, Broyden [1970], Davidon [1959] and Fletcher, Powell [1963].

The multi-model algorithms discussed above are for a special case of problem (1.1), the **nonlinear least squares problem :**

$$\begin{aligned} &\text{minimize } f(x) = \frac{1}{2} \| r(x) \|_2^2 \\ &r : \mathbb{R}^n \to \mathbb{R}^m \quad . \end{aligned} \tag{1.2}$$

These algorithms are among the best available methods for solving (1.2), yet little is known concerning their theoretical properties. Furthermore, the literature focuses to some extent on specific models introduced rather than developing the concept of multi-model algorithms fully, and little consideration is given to using multi-model algorithms for other special cases or for general unconstrained optimization. The lack of a theoretical framework is particularly unacceptable in light of the observation that other methods in the literature can be considered to be multi-model algorithms, even though not formulated explicitly as such. Typical of these are the many algorithms that ''reset'' or ''refresh'' a Hessian approximation under certain conditions. These conditions are typically defined arbitrarily, or at best determined heuristically.

The purpose of this thesis is twofold. First, the concept of using multiple models for general unconstrained minimization is dissected at length to clarify both the structure and goals of such algorithms. We classify existing methods using the framework thus constructed, discuss how well they satisfy our goals, and point out new ways of utilizing multiple models. Second, we develop a global convergence theory for trust region implementations of a large class of multi-model algorithms. This theory is surprisingly general, both in terms of the small number of restrictions on allowable model switching logic and in terms of the number of options allowed in the trust region logic. We can even include nonquadratic models in a natural way.

Throughout this paper, we use the following definitions and notation. The function to be optimized is $f(x)$, its gradient is $g(x)$, and its Hessian is $H(x)$. The residual vector of the nonlinear least squares problem is called $r(x)$, and the Jacobian of r is denoted $J(x)$. Depending on context, x^* denotes a local minimizer of f or a point satisfying $g(x)=0$. Our iterate at step k is denoted x_k, and a step away from x_k is called p, p_k, or p_k^i. The expressions g_k , H_k ,r_k , and J_k are shorthand for $g(x_k)$, $H(x_k)$, $r(x_k)$, and $J(x_k)$ respectively. An approximation to $H(x_k)$ is denoted by B_k or B_k^i. The Euclidean norm of a vector $v \in \mathbb{R}^n$ is denoted by $\| v \|$. For a matrix $B \in \mathbb{R}^{n \times n}$, $\| B \|$ refers to the operator norm induced by the Euclidean vector norm, while $\| B \|_F$ refers to the Frobenius norm of B.[4] DGW [1981] refers to references Dennis, Gay, Welsch [1981 a,b].

In actual implementation, modifications must be added to algorithms such as A.1.2 and A.1.3 to make them practical. The most significant changes are designed to aid the algorithm if the initial iterate is far from the solution. This can be done by adjusting the length of the step just computed (a *line search* procedure) or by imposing an adjustable constraint on the model at each iteration (a *trust region* procedure). A complete description of these procedures must be deferred to later chapters, but a rough sketch of the structure of Algorithm A.1.3 with the inclusion of

[4] The Frobenius norm of $B \in \mathbb{R}^{n \times n}$ is defined by $\| B \|_F = \left[\sum_{i,j} b_{i,j}^2 \right]^{1/2}$.

a trust region strategy is as follows.

Algorithm A.1.4 : A Multi-Model Algorithm with Trust Region Logic

```
0) Initialize.
1) Repeat until convergence :
    2) Select a model from two or more possibilities, and choose a trust radius
       $\Delta$.
    3) Compute an approximate local minimizer $\hat{x}$ for this model subject to
       $|| \hat{x} - x_k || \leq \Delta$.
    4) Either accept $\hat{x}$ as $x_{k+1}$ and proceed, or
             return to 2) and try another model and/or trust radius.
5) End .
```

1.2. Examples

In this section we present examples of situations where an algorithm using several models should be considered. These examples are included primarily for motivation at this point since detailed descriptions of implementations must be postponed until more notation is established.

There are several difficulties that a user must face in choosing from an available set of algorithms to solve a typical problem and that an expert must face in designing an algorithm for a specific type of problem. Four of these difficulties can be addressed by a multiple model algorithm. The first such difficulty is the identification of the type of problem. The second is the proper utilization of models which perform inconsistently as an iteration progresses. The third difficulty is the incorporation of newly derived models with unknown or unproven properties. A fourth difficulty is that of deciding between a mediocre model with low overhead and a more expensive high performance model. The first three difficulties are illustrated vividly by the nonlinear least squares problem.

Example 1.1 : Nonlinear Least Squares (NLS)

When encountered in the real world, an NLS problem falls into one of three categories : the zero-residual case, the small residual case, and the large residual case. In the zero residual case, $r(x^*)$ is zero. The other two cases lack a precise definition, but a problem is considered to be a small residual case if $r(x^*)$ is ''small'' or r is ''almost linear'' at x^*, and considered to be large residual otherwise.

Common methods for solving NLS problems have been the Gauss-Newton algorithm, the Levenberg-Marquardt algorithm (Levenberg [1944], Marquardt [1963]), and any of the standard secant algorithms for unconstrained optimization. Both the Gauss-Newton and Levenberg-Marquardt algorithms use

$$\phi(\ x+p\ ) = \frac{1}{2}\ \| J(x)p + r(x) \|_2^2 \tag{1.3a}$$

$$= f(x) + p^t J(x)^t r(x) + \frac{1}{2} p^t J(x)^t J(x) p \tag{1.3b}$$

as the local model, while a secant algorithm uses

$$\phi(\ x+p\ ) = f(x) + p^t J(x)^t r(x) + \frac{1}{2} p^t B\, p \quad . \tag{1.4}$$

These models differ only in how they approximate the Hessian of f. A secant method approximates the true Hessian $H(x)$ using some update formula, while the other two use the so-called *Gauss-Newton Hessian* : $J(x)^t J(x)$.

Let us assume for the moment that the secant update we are considering is the well known BFGS update. Then properly constructed trust region [5] algorithms based respectively on the two models have the properties shown in Table 1.

Now consider the "difficulties" mentioned above. First consider identification of problem type. We see that if a user wants to efficiently solve a problem using a single model algorithm, it is very important to know whether or not it is a zero

TABLE 1
RESULTS OF USING DIFFERENT
MODELS IN OTHERWISE IDENTICAL ALGORITHMS

Algorithm based on Gauss-Newton Hessian
Converges in one step on linear problems. Q-quadratically convergent on zero residual problems. Rapidly Q-linear on small residual problems. Very slow linear convergence on large residual problems. Globally convergent under mild assumptions on r.
Algorithm based on BFGS Hessian
Locally Q-superlinearly convergent for any residual size. More robust than an algorithm based on GN Hessian. Global convergence an open question. Does not take advantage of problem structure.

[5] To get the "global convergence" cited in Table 1 when using a trust region algorithm, we need only assume an upper bound on the norm of the Gauss-Newton Hessian. Line search algorithms require more restrictive assumptions, such as a full rank Jacobian.

residual case. This information is often not available in advance. Second, there is the problem of inconsistent performance. Algorithms based on the Gauss-Newton Hessian often outperform secant methods for several iterations even in relatively large residual cases.[6] Third, the structure of the NLS problem lends itself to the construction of alternative models. Most of these alternatives are generated by different approximations to the second order term $S(x)$, where

$$\begin{aligned} S(x) &\equiv H(x) - J(x)^t J(x) \\ &= \sum_{i=1}^{m} r_i(x) \nabla_x^2 r_i(x) \end{aligned} \tag{1.5}$$

At least *twenty* different ways for doing this exist in the literature. Some, such as DGW [1981], approximate S explicitly, while others, such as Nazareth [1980, 1983] or Al-Baali, Fletcher [1985] use the structure to treat S as an implicit part of B_k without explicitly carrying along an approximation to it. Further complications are introduced by some authors, such as schemes for "sizing" (DGW [1981]) or "scaling" (Oren, Luenburger [1974]) S to allow it to adapt to local conditions more rapidly. Furthermore, new models continue to appear in the literature. The theoretical properties of most of these models are not known. Even if analysis of some of these models is possible, the existing schemes differ from one another sufficiently that it is unlikely that this analysis would apply to the whole group.

[6] One potential explanation of this behavior involves the number of iterations required for a secant method to build up second derivative information. Additionally, the second order information inserted into B_k by the secant method may, in some cases, become quickly outdated due to rapid changes to H over the course of several long initial steps or in the course of rapid convergence to a zero residual solution. The Gauss-Newton Hessian, although incomplete, bases itself only on information at the current point.

Despite this lack of analysis, many of these schemes perform well and have been used successfully for years. However, the idea of trying out a code for years to see whether it stands the test of time seems inferior to proving that it will work when originally coded. We show in this thesis that a multi-model algorithm can be designed using one model with known properties and another model having totally unknown properties in such a way that global convergence is retained. This removes some of the urgency about having a full analysis of a new model before widespread use.

Lest the reader decide that multi-model algorithms are only attractive for nonlinear least squares problems, let us consider examples of multi-model algorithms for solving general unconstrained problems.

Example 1.2 : Constant Matrix for Model Hessian versus the Newton Model

The evaluation of an exact Hessian H_k is frequently prohibitively expensive. Using a fixed approximation B does away with this expense and substantially reduces linear algebra overhead, but may converge unacceptably slowly. The following multi-model algorithm is a compromise between the two extremes.

Algorithm A.1.5 : Newton Model versus Constant model

```
0) Initialize.
1) Repeat until convergence :
    2) Set $B_k = B$ .
    3) Set $x_{k+1}$ to an approximate local minimizer of the model.
    4) If $x_{k+1}$ is unsatisfactory, then
        If $B_k = B$, then set $B_k = H(x_k)$ and go to 3) ,
                Else reduce the trust radius and go to 3).
    5) Adjust the trust radius and proceed.
6) End .
```

Many other compromises are possible, of course. This particular version always tries the inexpensive model first, but quickly abandons it whenever faced with a trust radius reduction. An alternative would be to abandon it only if several such reductions are necessary, or to use another test entirely. Yet another alternative is to replace ``$B$'' in the algorithm by ``B_{k-1}'' so that rather than initially trying a fixed default model at each iteration, the algorithm initially tries the most recently computed exact Hessian. This algorithm and most of its variations are shown to be globally convergent in Chapter 4.

Example 1.3 : Secant Models and Full Newton Models

A more useful example is to apply a multi-model algorithm such as A.1.5 to a secant model and a full Newton model. Algorithms are already in existence which normally use a secant method because of its low expense, but which ``refresh'' this model to the exact Hessian occasionally. The theory in Chapters 5 and 6 show how this may be done in a globally convergent manner.

1.3. Synopsis In the remainder of this thesis, we are concerned primarily with establishing global convergence for as wide a class of algorithms as possible. Chapter 2 presents preliminary definitions and reviews single model trust region methods and results. Chapter 3 presents in more detail the goals one should keep in mind when designing a multi-model algorithm and gives examples of algorithms which fail to meet these goals. Chapter 4 presents a general trust region algorithm for model switching and establishes a form of global convergence based on the assumption that all the models are sufficiently well behaved. Specifically, we extend the global convergence theory of Schultz, Schnabel, and Byrd [1985] to allow the inclusion of any finite number of alternative quadratic models with uniformly bounded curvature. Chapter 5 presents a form of the algorithm which is slightly more restrictive in the choices allowed for model switching, but which allows looser conditions on the behavior of the models. The central feature of this class of algorithms is the test used in NL2SOL to distinguish between models. In Chapter 6 we present methods of exploiting these looser requirements to produce algorithms which are globally convergent if at least one of the models is sufficiently well behaved. The global convergence theory introduced in Chapter 5 and developed fully in Chapter 6 is therefore a significant and comprehensive extension of standard trust region theory to the multi-model case. Chapters 5 and 6 each present a slight variation of NL2SOL which is globally convergent under different assumptions on the models. Chapter 7 presents an illuminating example concerning local convergence rates of multi-model algorithms and discusses future work to be done in this area.

CHAPTER 2

Trust Regions Methods for Single Model Algorithms

2.1. Introduction

The two major strategies used for improving the global behavior of optimization algorithms are *line search* methods and *trust region* methods. In this thesis, we will restrict ourselves to trust region methods. Although line search methods generally require less code complexity, trust region methods seem more appropriate for a multi-model context. Theory developed for trust region methods can often be applied directly to line search methods. Trust region theory requires slightly less restrictive conditions on the models, and trust region methods extend more naturally than line search methods to models that are not quadratics with positive definite Hessians.

In the following sections, we present briefly the background necessary for the rest of this thesis. The first section deals with elementary definitions and some properties of functions in $\mathbb{R}^n$ that will be used in later chapters. The remaining sections review current trust region theory and practice for single model algorithms.

Trust region algorithms, particularly when applied to multiple models, involve a fairly extensive amount of notation. The reader is encouraged to make liberal use of Appendix A, which includes a brief glossary of the most important abbreviations, acronyms, symbols, and parameters used in this thesis.

2.2. Preliminaries

2.2.1. Models

Consider the **objective function** $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$ and let x_k be a vector in $\mathbb{R}^n$. A **model** of f with index i is denoted by

$$\phi_k^i(x) : \Theta_k^i \rightarrow \mathbb{R}^1 , \tag{2.1a}$$

where Θ_k^i is a subset of $\mathbb{R}^n$ containing x_k. We also define an associated function

$$pred_k^i(p) = f(x_k) - \phi_k^i(x_k+p) \tag{2.1b}$$

and call this the **predicted reduction**. Notice that $pred_k^i$ is expressed in local coordinates about the point x_k. We also define the **actual function reduction** as

$$ared_k(p) = f(x_k) - f(x_k+p) , \tag{2.1c}$$

so that $pred_k^i$ is intended to approximate $ared_k$. For convenience, we will usually refer to $pred_k^i$ as the model rather than ϕ_k^i.

For the rest of this chapter, we will assume that a single quadratic model is computed at each iteration k and drop the superscripts from ϕ_k^i and $pred_k^i$. For $x_k \in \mathbb{R}^n$ and a symmetric matrix B_k, **a quadratic model in standard form** refers to

$$\phi_k(x) = f(x_k) + g(x_k)^t (x-x_k) + \frac{1}{2}(x-x_k)^t B_k (x-x_k) \tag{2.2a}$$

or the associated function

$$pred_k(p) = -g_k^t p - \frac{1}{2} p^t B_k p \quad . \tag{2.2b}$$

The matrix B_k is said to be the **model Hessian.**

Even a partial treatment of the models commonly used in optimization is beyond the scope of this thesis. Although not completely necessary, a basic familiarity on the part of the reader with such topics as finite difference approximations of gradients and Hessians, secant methods, and elementary numerical linear algebra is strongly recommended. A concise yet thorough introduction to these subjects can be found in Dennis and Schnabel [1983].

2.2.2. Elementary Notation

We make the following definitions.

Definition 2.1 : The sequence $\{x_k\}$ in $\mathbb{R}^n$ is said to **converge** if there exists x^* such that $\lim_{k \to \infty} \| x_k - x^* \| = 0$. Furthermore, this is denoted by $x_k \to x^*$.

Definition 2.2 : The sequence $\{x_k\}$ in $\mathbb{R}^n$ is said to be **first order stationary point convergent** if $g_k \to 0$. Henceforth this will be denoted by **FOSPC** for brevity. FOSPC will also refer to an *algorithm* that generates sequences of iterates $\{x_k\}$ that are FOSPC.

Definition 2.3 : The sequence $\{x_k\}$ in $\mathbb{R}^n$ is said to be **weak first order stationary point convergent** if $\liminf_{k \to \infty} \| g_k \| = 0$. This property will also be denoted by the abbreviation **WFOSPC.**

Notice that $\{x_k\}$ is WFOSPC if and only if some *subsequence* of $g_k \to 0$.

Definition 2.4 : A set $\Omega \in \mathbb{R}^n$ is said to be **convex** if for every pair of vectors $x, y \subset \Omega$ and every $\xi \in [0,1]$ it follows that $\xi x + (1-\xi) y \in \Omega$.

Definition 2.5 : A sequence $\{x_k\}$ is said to have the **descent property** with respect to f if $f(x_{k+1}) \leq f(x_k)$ for all $k \geq 1$.

Definition 2.6 : Given a point $x_1 \in \mathbb{R}^n$, the set $L(f, x_1) \equiv \{ x : f(x) \leq f(x_1) \}$ is called the **level set** of f at x_1. Furthermore, the closed convex set

$\hat{L}(f, x_1) \equiv \{x : x = \xi y + (1-\xi) z, \ \xi \in [0,1], \ y, z \in L(f, x_1) \}$
is said to be the **convex hull** of $L(f, x_1)$.

Definition 2.7 : Given a convex set $\Omega \subset \mathbb{R}^n$, a function $f : \Omega \to \mathbb{R}^1$ is said to be **convex** if for every pair of vectors $x, y \in \Omega$ and every $\xi \in [0,1]$ it follows that $f(\xi x + (1-\xi) y) \leq \xi f(x) + (1-\xi) f(y)$.

2.2.3. Some Multivariant Calculus Background

In future sections we will make use of the following definitions and results. Lemmas 2.1 through 2.5 are adapted from Lemmas 4.1.2, 4.1.5, and 4.1.9 of Dennis and Schnabel [1983].

Definition 2.8 : For a given $x \in \mathbb{R}^n$ and nonzero $p \in \mathbb{R}^n$, the **directional derivative** of f at x in the direction p is

$$\frac{\partial f}{\partial p}(x) \equiv \lim_{\varepsilon \to 0} \frac{1}{\varepsilon}(f(x+\varepsilon p)-f(x))$$

if the limit exists. The **second directional derivative** of f at x in the direction p is

$$\frac{\partial^2 f}{\partial^2 p}(x) \equiv \lim_{\varepsilon \to 0} \frac{1}{\varepsilon}(\frac{\partial f}{\partial p}(x+\varepsilon p)-\frac{\partial f}{\partial p}(x))$$

if the limit exists.

Lemma 2.1 : *If the function $f : \mathbb{R}^n \to \mathbb{R}^1$ is continuously differentiable with gradient g in an open set containing the point x, then for any nonzero $p \in \mathbb{R}^n$, the directional derivative of f at x in the direction p exists and has the value $g(x)^t p$.*

Lemma 2.2 : *If the function $f : \mathbb{R}^n \to \mathbb{R}^1$ is twice continuously differentiable with Hessian H in an open set containing the point x, then for any nonzero $p \in \mathbb{R}^n$, the second directional derivative of f at x in the direction p exists and has the value $p^t H(x)p$.*

We will often use the second directional derivative to characterize the shape of a function.

Definition 2.9 : For a nonzero vector $p \in \mathbb{R}^n$, $\dfrac{p^t H(x)p}{p^t p}$ is said to be the **curvature** of f in the direction p.

Definition 2.10 : Any vector $p \in \mathbb{R}^n$ that satisfies $p^t H(x)p < 0$ is said to be a

direction of negative curvature for the function f at the point x.

Definition 2.11 : A symmetric matrix $B \in \mathbb{R}^{n \times n}$ is said to be **positive definite** if $x^t Bx > 0$ for every nonzero $x \in \mathbb{R}^n$, **positive semidefinite** if $x^t Bx \geq 0$ for every $x \in \mathbb{R}^n$, **indefinite** if $x^t Bx > 0$ for some $x \in \mathbb{R}^n$ and $y^t By < 0$ for some $y \in \mathbb{R}^n$, and **negative definite** if $x^t Bx < 0$ for every nonzero $x \in \mathbb{R}^n$.

Definition 2.12 : A point $x^* \in \mathbb{R}^n$ is said to be a **saddle point** of the function f if $g(x^*) = 0$ and $H(x^*)$ is indefinite.

Lemma 2.3 : *If the function $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$ is continuously differentiable in an open convex set $\Omega \subset \mathbb{R}^n$, then for any x, $x + p \in \Omega$*

$$f(x+p) - f(x) = \int_0^1 g(x + \lambda p)^t p \, d\lambda \quad . \tag{2.3}$$

Lemma 2.4 : *If the function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is continuously differentiable in an open convex set $\Omega \subset \mathbb{R}^n$, then for any $x, x + p \in \Omega$,*

$$F(x+p) - F(x) = \int_0^1 \nabla F \, (x + \lambda p)^t p \, d\lambda \tag{2.4}$$

where ∇F is the transpose of the Jacobian of the function F.

Lemma 2.5 : *If the function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is continuously differentiable in an open convex set $\Omega \subset \mathbb{R}^n$, then for any $x, x + p \in \Omega$,*

$$\| F(x+p) - F(x) \| \leq \|p\| \int_0^1 \| \nabla F \, (x + \lambda p) \| \, d\lambda \quad . \tag{2.5}$$

Lemma 2.6 : *If the function $f: \mathbb{R}^n \to \mathbb{R}^1$ is twice continuously differentiable in an open convex set $\Omega \subset \mathbb{R}^n$, then for any $x, x+p \in \Omega$*

$$f(x)-f(x+p) = -g(x)^t p - \int_0^1 p^t H(x+\xi p)p(1-\xi)\,d\xi \quad . \tag{2.6}$$

Furthermore, if $x_k, x_k+p_k \in \Omega$ and if $pred_k$ is a quadratic model in standard form, we have

$$ared_k(p_k)-pred_k(p_k) = \int_0^1 (1-\xi)(p_k)^t\,(B_k - H(x_k+\xi p_k))\,(p_k)\,d\xi \quad . \tag{2.7}$$

Proof : From Lemma 2.3 we have that

$$f(x)-f(x+p) = -\int_0^1 g(x+\lambda p)^t p\,d\lambda \quad .$$

From Lemma 2.4 we have that

$$\begin{aligned} g(x+\lambda p) &= g(x) + \int_0^1 \nabla g(x+\tau\lambda p)^t \lambda p\,d\tau \\ &= g(x) + \int_0^\lambda H(x+\xi p)p\,d\xi \ . \end{aligned}$$

Combining the two gives

$$\begin{aligned} f(x)-f(x+p) &= -\int_0^1 \left[g(x)+\int_0^\lambda H(x+\xi p)p\, d\xi \right]^t p\, d\lambda \\ &= -\int_0^1 g(x)^t p\, d\lambda - \int_0^1\int_0^\lambda p^t H(x+\xi p)p\, d\xi d\lambda \\ &= -g(x)^t p - \int_0^1 \left[\int_\xi^1 p^t H(x+\xi p)p\, d\lambda \right] d\xi \\ &= -g(x)^t p - \int_0^1 (1-\xi)p^t H(x+\xi p)p\, d\xi \end{aligned} \tag{2.8}$$

which proves the first part of the lemma. We then have

$$\begin{aligned} pred_k(p_k) &= -g(x_k)^t p_k - \frac{1}{2}(p_k)^t B_k (p_k) \\ &= -g(x_k)^t p_k - (p_k)^t B_k (p_k) \int_0^1 (1-\xi) d\xi \\ &= -g(x_k)^t p_k - \int_0^1 (1-\xi)(p_k)^t B_k (p_k) d\xi \end{aligned} \tag{2.9}$$

so that setting $x = x_k$ and $p = p_k$ and subtracting (2.9) from (2.8) establishes the second part of the lemma. □

2.3. The Trust Region Subproblem

Before considering the full trust region algorithm, we must first discuss what is known as the *trust region subproblem.*

Definition 2.13 : Given $\Delta_k > 0$, the **trust region subproblem** (denoted TRS) is

$$\underset{\|p\| \le \Delta_k}{\text{maximize}}\ pred_k(p)\ . \tag{TRS}$$

The ball $\{ p : \|p\| \le \Delta_k \}$ is known as the **trust region.**

An important practical point in trust region algorithms is the inclusion of a positive definite scaling matrix, D, which replaces the spherical trust region heretofore assumed with the hyperellipse

$$\| Dp \| \leq \Delta_k \quad . \tag{2.10}$$

Typically, D is a diagonal matrix. If D is constant, then its inclusion amounts to a simple change of variables $w = Dp$ that converts the problem defined by (2.10) back to the standard problem TRS with

$$pred_k(w) = -g_k^t D^{-1} w - \frac{1}{2} w^t D^{-1} B_k D^{-1} w$$

so that the trust region theory is unaffected. Other trust region methods use a sequence of scaling matrices $\{D_k\}$ instead of a single fixed D. Much of the existing global convergence theory can be extended to these methods provided both $\{D_k\}$ and $\{D_k^{-1}\}$ are uniformly bounded. An excellent reference to scaling matrices is Moré and Sorensen [1983]. For simplicity, we will take $D_k \equiv I$ throughout this thesis.

Computation of an exact solution to problem (TRS) is not practical except in certain special cases (such as when ϕ_k is convex and the constraint $\| p \| \leq \Delta_k$ is not binding). Consequently much of the literature deals with methods of computing approximate solutions to the problem TRS. A good introduction to these techniques is Dennis and Schnabel [1983]. Some excellent references for a more advanced treatment are Moré [1982], Moré and Sorensen [1982, 1983], Schultz, Schnabel, and Byrd [1985] and Steihaug [1981]. A very brief review of some of these methods follows.

2.3.1. Optimal Locally Constrained Steps

Some methods approximate a solution to problem TRS by computing an exact solution to a "nearby" problem which is easier to solve. In this context, a convenient "nearby" problem is to maximize $pred_k$ subject to a different constraint (say, $\|p\| \leq \tilde{\Delta}_k$ where $\tilde{\Delta}_k \approx \Delta_k$). We refer to such methods as optimal locally constrained (or OLC) methods. The reader is referred to Moré and Sorensen [1983] for the history, theory and practice of these methods, but for the purposes of this discussion we only need to point out a few properties.

OLC steps (that is, the approximate solutions to problem TRS generated by an OLC method) do not necessarily lie in the trust region, but are allowed to lie in the region $\{p : \|p\| \leq (1+\sigma_1)\Delta_k\}$ where $\sigma_1 \in (0,1)$. NL2SOL uses $\sigma_1 = 0.1$, but algorithms which use values such as $\sigma_1 = 0.5$ are also found in the literature. A minimal requirement for an OLC step is that the predicted reduction for this step must be at least a given fraction of the predicted reduction of the exact solution to problem TRS. That is, if p_k^* is the solution to problem TRS and p_k is an approximation generated by an OLC method, there exists $c_2 > 0$ independent of k such that

$$pred_k(p_k) \geq c_2 pred_k(p_k^*) \ . \tag{2.12}$$

Moré and Sorensen [1983] present an algorithm that will generate a p_k satisfying (2.12) in a finite number of steps. Furthermore, computational experience indicates that on the average less than two steps (each involving the Cholesky[1] factorization of an $n \times n$ matrix) are required to generate p_k satisfying (2.12). The algorithm does not require that ϕ_k be convex, and in fact makes use of directions of negative curvature to inhibit convergence to saddle points and to choose directions of rapid decrease.

2.3.2. Restricted Subspace Methods

Another approach to approximating p_k^* is to consider only the values of p in a given subspace $\Omega_k \subset \mathbb{R}^n$.

Definition 2.14 : Given a scalar $\Delta_k > 0$ and a subspace $\Omega_k \subset \mathbb{R}^n$, the **restricted subspace trust region subproblem** is

$$\begin{aligned} &\underset{p \in \Omega_k}{\text{maximize}}\ pred_k(p) \\ &s/t \qquad \|p\| \leq \Delta_k \quad . \end{aligned} \tag{2.13}$$

A restricted subspace method will compute an approximate solution to (2.13), and use this as approximation to p_k^*. If Ω_k is of small dimension when compared to $\mathbb{R}^n$, then such a method can subsequently reduce overhead. Typically, Ω_k is chosen to be the two dimensional space spanned by $-g_k$ (the direction of steepest descent) and $-(B_k)^{-1} g_k$ (the quasi-Newton direction). For positive definite B_k, these two directions are natural because they represent p_k^* in the two limiting cases $\Delta_k \to 0$ and $\Delta_k \to \infty$, respectively. Furthermore, solving (2.13) over a two dimensional subspace is equivalent to solving a fourth order polynomial in one unknown, which is a

[1] See, for example, Golub and Van Loan [1983], Section 5.2.

computationally tractable problem. A disadvantage of most of these methods is that they do not ensure (2.12), but instead lead to a weaker condition: that the predicted reduction is at least a fraction of the predicted reduction at the solution of (2.13). When $g_k \in \Omega_k$, solving (2.13) implies that there exists $c_1 > 0$ independent of k such that

$$pred_k(p_k) \geq c_1 pred_k(p_k^c) \tag{2.14}$$

where p_k^c solves

$$\begin{aligned} &\underset{p=\alpha g_k}{\text{maximize}}\ pred_k\,(p) \\ &s/t \quad ||p|| \leq \Delta_k \quad . \end{aligned} \tag{2.15}$$

We refer to p_k^c as the Cauchy step.

A much more complete description of restricted subspace methods can be found in Moré [1982].

2.3.3. Dogleg Methods.

A third approach to approximating p_k^* can be motivated in several ways.

Definition 2.15 : Let Ω_k be a subspace of $\mathbb{R}^n$ with $g_k \in \Omega_k$. Consider a curve $p(\alpha)$ in Ω_k with α defined on [0,1]. This curve is said to be a **generalized dogleg** if it satisfies the following conditions.

(a) $p(0)=0$.

(b) g_k is tangent to $p(\alpha)$ at $\alpha=0$.

(c) $\|p(\alpha)\|$ is strictly monotone increasing in α.

(d) $pred_k(p(\alpha))$ is strictly increasing in α over [0,1].

(e) $p(1)$ is a global maximizer of $pred_k$ over Ω_k if B_k is positive semidefinite, else $\|p(1)\| \geq \Delta_k$.

Loosely speaking, a generalized dogleg is a curve in $\mathbb{R}^n$, starting at the origin, proceeding in a path that increases $pred_k$, and terminating at the global maximizer of $pred_k$ (if such a maximizer exists). Thus the problem

$$\underset{\substack{\alpha\in[0,1] \\ s/t \|p(\alpha)\| \leq \Delta_k}}{\text{maximize}}\ pred_k(p(\alpha)) \tag{2.16}$$

has two possible solutions $p(\alpha^*)$:

$$\alpha^* = 1\ ,\ p(\alpha^*) = p_k^* \tag{2.17}$$

or

$$\alpha^* \in (0,1)\quad ,\quad \|p(\alpha^*)\| = \Delta_k\quad . \tag{2.18}$$

The first occurs when $p(1)$ lies within the trust region. For positive definite B_k, $p(1)$ is simply the *quasi-Newton step*, $-B_k^{-1}g_k$, denoted p_k^n. The second case occurs when no global maximizer exists, or when it is outside the trust region. Here, $p(\alpha^*)$ is simply the point of intersection of $p(\alpha)$ with the surface of the trust region.

Expressed in this fashion, problem (2.16), is clearly meant to be similar to the restricted subspace problem (2.13), but with the potential of requiring even less computational overhead. The original dogleg was suggested by Powell [1970 a,b], who used the curve

$$p(\alpha) = \begin{cases} 2\alpha p_k^c & \alpha < \frac{1}{2} \\ 2(1-\alpha)p_k^c + (2\alpha - 1)p_k^n & \alpha \geq \frac{1}{2} \end{cases} \tag{2.19}$$

This curve consists of two line segments. The first starts at $p = 0$ and proceeds to the Cauchy step p_k^c and the second joins the Cauchy step with the quasi-Newton step p_k^n. The only matrix factorization needed to define $p(\alpha)$ and compute $p(\alpha^*)$ is the original factorization used to compute p_k^n. For secant methods that update B_k in factored form, the overhead drops even further to order n^2. Further savings can be realized when the problem TRS must be resolved with a different Δ_k. OLC methods must resolve the problem almost from scratch, entailing even more matrix factorizations. Dogleg algorithms need only compute the new point of intersection of $p(\alpha)$ with the surface of the new trust region.

Dennis and Mei [1979] present a *double dogleg* algorithm similar to Powell's dogleg method, but which uses three line segments in $p(\alpha)$. Steihaug [1981] develops a multi-segmented dogleg method, based on the conjugate gradient method. This technique is natural and attractive in that successive conjugate gradient steps maximize $pred_k$ over a sequence of successively expanding Krylov subspaces. Dogleg procedures that take advantage of indefiniteness are presented by Schultz, Schnabel and Byrd [1985] and Steihaug [1981].

Although dogleg algorithms have the advantage of reduced overhead, they may generate steps inferior to those produced by OLC algorithms. The ideal dogleg would follow the curve of solutions to problem TRS as Δ_k increases from zero. This may differ considerably from the path defined by a dogleg or double dogleg.

However, computational experience with dogleg algorithms suggests that any decrease in performance is not significant enough to outweigh the dogleg's advantages. At present the choice between dogleg and OLC methods seems to be a matter of taste.

Dogleg algorithms generate steps that satisfy slightly different conditions from OLC algorithms. Taking the Dennis-Mei double dogleg (with B_k assumed to be positive definite) as an example, the generated step p_k satisfies equations (2.14) and (2.15) rather than the stronger (2.12). Furthermore, we have either

$$p_k = -(B_k)^{-1} g_k \quad \text{and} \quad \| p_k \| < \Delta_k \tag{2.20}$$

or

$$\| p_k \| = \Delta_k \ . \tag{2.21}$$

For a more extensive survey of dogleg type algorithms, including material on preconditioning, the reader is again referred to the excellent article Moré [1982].

2.4. Trust Region Algorithms

Different authors have created many different versions of trust region algorithms. Although the central features are the same, some of the details differ sufficiently to cause a certain amount of fragmentation in the theory as presented in the literature. Proofs of most results are specific to a given implementation, and it is often not immediately clear whether a slightly different implementation is equivalent. Much of this is the result of sacrificing generality for clarity; including the possibility of many options may obscure more important points about an algorithm or proof. In this thesis we present simplified sample algorithms when clarity is desired but otherwise present theory for algorithms written to allow a large number of different implementations.

We now present three representative trust region formulations.[2]

Algorithm A.2.1 : A Minimal Version of a Trust Region Algorithm

```
0) Let $x_1 \in \mathbb{R}^n$ and $\Delta_1 > 0$ be given.
1) Repeat until convergence :
    2) Compute an approximate solution $p_k$ to problem TRS.
    3) If $\frac{ared_k(p_k)}{pred_k(p_k)} < \frac{1}{100}$, then set $\Delta_k = \Delta_k / 4$ and return to 2.
    4) Set $x_{k+1} = x_k + p_k$ , $\Delta_{k+1} = \Delta_k$.
    5) If $\frac{ared_k(p_k)}{pred_k(p_k)} \geq \frac{1}{2}$ then set $\Delta_{k+1} = 2 \times \Delta_k$.
6) End.
```

This algorithm should be a clear demonstration of the central idea. The *trust radius* Δ_k is adjusted at each iteration so that the model and function values retain at least a minimal amount of agreement over the trust region. The measure used for judging the agreement between model and function is the ratio of the actual reduction

[2] Trust region algorithms generally include several parameters that differ from implementation to implementation. The sample formulations presented in this chapter do not include the full range of parameters we will use in Chapters 4 and 5. In order to keep our notation consistent, the subscripts used with parameters in this chapter are defined to match the subscripts needed in later chapters. The reader is referred to Appendix A for typical values assigned to these parameters.

to predicted reduction for the last step. No step is accepted unless this ratio achieves a certain minimum value. We will sometime refer to a step as ``satisfactory'' or ``unsatisfactory'' based on this criterion.

The approximate solution to problem TRS can be computed by the methods of the last section, and is required to satisfy (2.12) or (2.14) as well as the matching conditions on the norm of p_k with respect to Δ_k.

A well-known result (see, for example, Powell [1975]) is that either (2.12) or (2.14) is sufficient to imply that there exists $c > 0$ independent of k for which

$$pred_k(p_k) \geq c \, \| g_k \| \min\{ \Delta_k , \frac{\| g_k \|}{\| B_k \|} \} \ . \tag{2.22}$$

This in turn implies that the sequence of iterates produced by the algorithm is FOSPC provided $\{B_k\}$ is uniformly bounded and f satisfies certain mild conditions.

Now that we have demonstrated the basic elements, let us examine the formulation presented by Powell [1984].

Algorithm A.2.2 : Trust Region Algorithm / Powell [1984]

```
0) Let $0<\gamma_0\le\gamma_1<1\le\gamma_3$, $0<\eta_3<1$, $x_1\in \mathbb{R}^n$ , $\Delta_1>0$ be given.

1) Repeat until convergence :

    2) Compute $p_k$ satisfying (2.22) and either (2.20) or (2.21).

    3) If $ared_k(p_k)>0$, then set $x_{k+1}=x_k+p_k$ and update $B_k$,
       Else set $x_{k+1}=x_k$.

    4) If $ared_k(p_k)\ge\eta_3\, pred_k(p_k)$ then set $\Delta_{k+1}\in[\,\|p_k\|\,,\gamma_3\|p_k\|\,]$,
       Else set $\Delta_{k+1}\in[\gamma_0\|p_k\|\,,\gamma_1\|p_k\|\,]$.

5) End.
```

Powell [1984] shows that this algorithm produces iterates that are WFOSPC under mild conditions on f provided there exist constants β_2 and $\bar{\beta}_2$ for which

$$\|B_k\|\le\beta_2+\bar{\beta}_2 k. \tag{2.23}$$

This algorithm differs from A.2.1 in several particulars. A.2.2 updates the trust radius based on $\|p_k\|$ rather than Δ_k as in A.2.1. The steps computed for A.2.2 must satisfy (2.20) or (2.21), while A.2.1 allows the more relaxed bound $\|p_k\|\le(1+\sigma_1)\Delta_k$.

More importantly, Powell allows the acceptance of a step if $\dfrac{ared_k(p_k)}{pred_k(p_k)}>0$ rather than demanding that the ratio be greater than a positive constant.

Schultz, Schnabel and Byrd [1985] give an abstract version of a trust region algorithm that admits most of the trust region logic in use today. Their algorithm, which is FOSPC when $\{B_k\}$ is uniformly bounded and f satisfies mild conditions, is defined as follows.

Algorithm A.2.3 : Trust Region Algorithm / SSB [1985]

```
0) Let $\gamma_1, \eta_1, \eta_2 \in (0,1), x_1 \in \mathbb{R}^n, \Delta_0 > 0$ be given.
1) Repeat until convergence:
    2) Find $\Delta_k$ and compute $p_k = p_k(\Delta_k)$ satisfying $\| p_k \| \leq \Delta_k$ and
        a) $\frac{ared_k(p_k)}{pred_k(p_k)} \geq \eta_1$ and
        b) Either $\Delta_k \geq \Delta_{k-1}$, or
           $\Delta_k \geq \| B_{k-1}^{-1} g_{k-1} \|$ with $B_{k-1}$ positive definite or
           For some $\Delta \leq \frac{1}{\gamma_1} \Delta_k$ ,
           $\frac{ared_k(p_k(\Delta))}{pred_k(p_k(\Delta))} < \eta_2$ or $\frac{ared_{k-1}(p_{k-1}(\Delta))}{pred_{k-1}(p_{k-1}(\Delta))} < \eta_2$ .
    3) Let $x_{k+1} = x_k + p_k$ and $k = k+1$ .
4) End.
```

Rather than directly specifying a technique to be used for computing steps p_k, Schultz, Schnabel, and Byrd present general conditions that steps must meet to be considered acceptable. These conditions therefore define the allowable range of options in the unspecified portions of the algorithm.

Much of our multi-model theory is a generalization of the trust region theory in this paper. We shall abbreviate all references to this paper for brevity, calling it SSB [1985]. Furthermore, the general algorithms we present will be abstract in the same sense as A.2.3; parts of the algorithm will be left undefined except for our specification of certain conditions the algorithm must obey.

CHAPTER 3

Motivation and Nomenclature for Multi-model Algorithms

3.1. Introduction

In this chapter we examine some classes of multi-model algorithms. Section 3.2 examines the motivations and principles we use in designing a multi-model algorithm. Section 3.3 introduces nomenclature for classifying multi-model algorithms into different categories, points out where the existing algorithms fall in such a classification scheme, and gives a preliminary discussion of the advantages and disadvantages of algorithms in these categories.

3.2. Motivation and Goals

Before examining the theory or implementation of a multi-model algorithm, we should closely examine what we hope to accomplish. Multi-model algorithms invariably require more computational overhead, storage, and code complexity. Thus it is imperative that any such algorithm be examined closely to see that it does at least *something* better than either of the corresponding single model algorithms. Several possibilities suggest themselves immediately. A multi-model algorithm may perform better in practice, it may have better theoretical global convergence properties than a corresponding single model algorithm, or it may have better theoretical local convergence properties.

Extensive testing of diverse multi-model algorithms for NLS problems has shown them to be often superior to single model algorithms, even for somewhat ad-hoc incorporation of the additional models. Given this practical success, we deal strictly with theoretical properties in this thesis. However, this theory must deal with practical algorithms, and thus we must discuss what properties a ''practical'' algorithm should have.

One necessary property is efficiency. Any procedure devised for selecting or switching models cannot add so much overhead that the algorithm is too expensive to run. The ''ideal'' solution would be an algorithm that selects the single best model at each iteration without any extra overhead or function evaluations. A ''worst case'' example might involve computing and testing trial steps for each possible model at each stage of the algorithm. Such an algorithm might be as follows.

Algorithm A.3.1 : Sample algorithm with excessive overhead

```
0) Initialize.
1) Repeat until convergence :
    2) Repeat until all models have been tried.
        Select a model.
        Compute a trial iterate for this model.
        Compute $f$ for this trial iterate.
    3) Choose the ``best'' iterate computed above. If it is ``unsatisfactory,''
        then reduce the trust radius and return to 2).
    4) Adjust the trust radius.
5) End.
```

Although this process reduces the number of external iterations -- the loop between 1) and 5) -- needed to achieve a given tolerance, it is clearly too inefficient to be useful unless the function evaluations and linear algebra involved are exceedingly inexpensive. Even if only two models are used and the trust radius is never decreased, this algorithm doubles the number of function evaluations needed to perform a given number of iterations. For more than two models, and with a typical frequency of trust radius reductions, both the number of function evaluations and the amount of numerical linear algebra needed can be a large multiple of the corresponding single model algorithm.

Another necessary goal is reliability. In looking at theoretical global or local properties, the best of all possible worlds would be a model switching algorithm that always behaves at least as well as would be expected for a single model algorithm using the "best" model. With this in mind, we often examine how well a proposed multi-model algorithm can handle the case when one of its models becomes arbitrarily bad, and we are most satisfied with algorithms that do not have their performance degraded by such models. As discussed in the examples of Section 1.2, such a multi-model algorithm allows one to include in production code models which may sometimes perform badly, or even entirely speculative models, without fear of disaster. For a SSB [1985] type single model trust region algorithm such as A.2.3, FOSPC can be obtained under mild conditions on f if the sequence of model Hessians is uniformly bounded. For a Powell [1984] type trust region algorithm such as A.2.2, the condition on the sequence of model Hessians is relaxed slightly, but the convergence result is weaker (WFOSPC rather than FOSPC). In a multi-model algorithm, we consider it quite reasonable to require that *one* model obey the stronger condition. Consequently, we put our algorithms in a framework similar to SSB [1985] to obtain the stronger results. References to ''global convergence'' should be interpreted as FOSPC unless otherwise stated. An algorithm is referred to as ''safe'' if we can specify conditions on the models which imply that the algorithm is FOSPC.

Many of the obvious ways to avoid arbitrarily bad models are incompatible with the goal of ''efficiency.'' Algorithm A.3.1 above certainly satisfies our goal of reliability but is not efficient in most cases. Another reliable algorithm is as follows. Let B be a given symmetric matrix.

Algorithm A.3.2 : Sample algorithm which is reliable but inefficient

```
0) Initialize.
1) Repeat until convergence :
    2) Set $B_k$ to be a secant update of $B_{k-1}$.
    3) If $|| B_k || > || B ||$ set $B_k$ to $B$.
    4) Compute a trial iterate for this model.
    5) If it is "unsatisfactory," then reduce the trust radius and return to 4).
    6) Adjust the trust radius.
7) End.
```

Now, if B is defined so that it properly reflects the scale of the problem over the entire range of iterates, this algorithm is both safe and efficient. However, if B does not reflect the scale of the problem, it can force the algorithm to use a highly inappropriate model even if the secant model is performing well. As an extreme example, consider using the identity matrix for B. Then whenever the norm of the secant update is greater than one, B_k is set to I and the algorithm reverts to the method of steepest descent.

In analyzing efficiency, the hierarchy usually assumed is that evaluating an exact Hessian is much more expensive than evaluating a gradient, which is much more expensive than evaluating a function, which is more expensive than calculating a trial step for a given model, which is more expensive than evaluating the predicted reduction for a trial step. A major feature of multi-model algorithms is that they can be tailored to specific cases within this hierarchy, or to other hierarchies altogether. For example, although Algorithm A.3.1 is quite inefficient under the standard hierarchy, it becomes reasonable if the expense of evaluating g_k is *extremely* large with respect to the expenses associated with computing steps and evaluating f. Other special cases may also make A.3.1 attractive. For certain classes of array processors and certain classes of objective functions, f can be simultaneously calculated for a large number of different x values at little more expense than a single function evaluation. In such a situation A.3.1 would be competitive. Flexibility in algorithm design is also needed to take advantage of tradeoffs between ‘‘expensive’’ and ‘‘inexpensive’’ models as in Examples 1.2 and 1.3. Algorithms specifically designed for such cases are important enough to warrant inclusion in this thesis. Thus it is imperative that our global convergence theory be flexible enought to allow such possibilities.

None of these properties or goals can be sacrificed in favor of the others if our theory is to be realistic. Chapters 4, 5, and 6 present algorithmic frameworks flexible enough to allow a wide range of different implementations for different situations while retaining both reliability and high efficiency.

3.3. Nomenclature for Multi-model Algorithms

At this point we introduce some more nomenclature. Most of it concerns the *switching system :* the scheme used to decide which models to use and how to use them. We define three different types of switching systems as follows.

Definition 3.1 : A **model selection** algorithm is one in which a single model is selected at the end of each iteration for use throughout the next iteration.

This approach is very amenable to analysis with standard global convergence theory since only one model is being used at each iteration. All Al-Baali, Fletcher [1985] and Nazareth [1980,1983] algorithms are of this type.

Definition 3.2 : A **model switching** algorithm is one in which two or more candidate models are available to the system at any iteration. The algorithm can use any or all of these models to compute and/or test steps, and it can change models during an iteration based on these computations.

NL2SOL (DGW [1981]) is an algorithm of this type, and all model selection algorithms are special cases of model switching algorithms.

Definition 3.3 : A **concurrent model** algorithm is one in which more than one model can be used simultaneously in a given iteration.

For example, a possible algorithm mentioned by Nazareth (but not tried) involved computing two steps -- each based on a different model -- and choosing the actual step to be some convex combination of the two steps.

We now discuss each of these types in detail and preview some of our future results.

3.3.1. Model Selection Algorithms

3.3.1.1. The Algorithm

Model selection algorithms are the most common type in the literature. Implementations are simple and reportedly highly successful. Furthermore, since only one model is used at each iteration, global results can be stated immediately about many of these algorithms if implemented consistently with one of the trust region frameworks of Chapter 2, and if the sequence of model Hessians selected is uniformly bounded. Consider the nonlinear least squares problem (NLS). Fletcher, Al-Baali, and Nazareth hybrid algorithms always deal with the Gauss-Newton model $J(x_k)^t J(x_k)$ (which we assume is bounded in k), and a model produced by a secant method. If there exists a uniform upper bound on the curvature of the models produced by the secant method, then most trust region algorithms are FOSPC no matter *how* the models are combined. However, the existence of such an upper bound is an open question. Furthermore, if we assume such a bound, we beg the question of what role model selection actually plays in helping the algorithm achieve FOSPC. The really interesting question is whether any of the tests in the literature can distinguish between a sequence of model Hessians which is uniformly bounded and a sequence which is unbounded. This question is answered in Section 3.3.1.3.

Algorithm A.3.3 is an example of a model selection algorithm written as a trust region method. Details not pertaining directly to the use of more than one model are suppressed for clarity.

Algorithm A.3.3 : Model Selection

```
0) Let $x_1 \in \mathbf{R}^n$ be given.
   Define an initial trial trust radius $\bar{\Delta}_1 > 0$, and a set of model Hessians $\{B_1^i\}$.
   Choose one of the models to be the current preference.
   Let $\bar{i}$ be the index of this model.

1) Repeat until convergence :

    2) Compute a trial step $p_k^{\bar{i}}$ that approximately solves (TRS) for the trial trust radius
       $\bar{\Delta}_k$ and current model with index $\bar{i}$ . Then either

           (a) Accept $\Delta_k = \bar{\Delta}_k$ and $p_k^{\bar{i}}$, or

           (b) Find a reduced trust radius $\Delta_k < \bar{\Delta}_k$ and compute a "satisfactory" $p_k^{\bar{i}}$ that
               approximately solves (TRS) for the currently preferred model.

       Set $p_k = p_k^{\bar{i}}$.

    3) Choose a trial radius $\bar{\Delta}_{k+1}$ for the next step.

    4) Perform any calculations required to make the models available at the next itera-
       tion, and select the model Hessian $B_{k+1}^{\bar{i}}$ to be used.

    5) Set $x_{k+1} = x_k + p_k$, $f_{k+1} = f_k$ , $g_{k+1} = g(x_{k+1})$, $k = k + 1$.

6) End.
```

Some of the model selection algorithms in the literature are implemented as line search methods, but we analyze them as if they were implemented using trust regions. Most consider only two models at each iteration. Algorithm A.3.3 allows for any number of quadratic models to be generated at each iteration, but only one is actually selected. The choice of the model to be used in the next iteration can be made before or after the models are updated or recalculated. In general, steps 3) - 5) can be done in any order desired.

Step 4) is phrased to allow the possibility of not actually forming a model unless it is needed in the next iteration.

3.3.1.2. Tests for Choosing Among Alternative Models

There are several schemes in the literature for choosing a model for the next iteration. We first consider the different tests used to evaluate the performance of a model and then demonstrate how the algorithms put these tests to use. Each test is based on computing some measure of how "good" a model is. This value can then be tested against a fixed constant to classify the model as "acceptable" or "unacceptable," or two models can be compared by computing the "goodness" measure for each model and comparing the values. Al-Baali and Fletcher [1985] suggest a **variational test** for the nonlinear least squares problem. This test calculates

$$V_k(B_k^i) = \| U^{-1} - (B_k^i)^{-1} \|_{F,U} \tag{3.1}$$

for a positive definite Hessian approximation B_k^i, where U is a BFGS type update of B_k^i and $\| * \|_{F,U}$ is the weighted Frobenius norm defined by $\| A \|_{F,U} \equiv \| U^{-1/2} A U^{-1/2} \|_F$. Values close to zero indicate a "good" model. The matrix U could actually be any symmetric positive definite matrix, but the above definition allows this test to be interpreted in a natural fashion. If the test is applied to the matrix B_k^i after it has been computed at the end of iteration $k-1$, it can be viewed as a measure of "how close" the matrix B_k^i is to satisfying the secant equation. However, to be meaningful when the model B_k^i being tested is itself a structured version of the BFGS secant model, the test must be done before the update, since otherwise the value of V_k for this model is automatically zero. Testing before the update leads to the interpretation that V_k measures "how close" the matrix B^i_{k-1} is to satisfying the secant equation, or (for the BFGS case) "how much the model will change." Fletcher and Al-Baali use as their two models the Gauss-Newton Hessian and a structured BFGS secant approximation. Although their implementation tests the Gauss-Newton model after it is generated and thus mixes the interpretations above, they report a great deal of success in practice. Their test does not require any extra function evaluations and can be implemented without excessive amounts of extra numerical linear algebra. It should thus be considered as a viable method for selecting a model for use in the next iteration. The major drawback from the general point of view is that there exist many models to which it is not applicable. Some of the structured models for nonlinear least squares do not necessarily remain positive definite, so the test cannot be applied.[1] Nonquadratic models are also untestable by this measure. Furthermore, it can only be used for model selection and not for model switching.

Other tests used by various authors compare the observed accuracy of different models for a given step. Given a test step $\hat{p}$, a **ρ test** calculates the ratio

$$\rho^i_k(\hat{p}) \equiv \frac{ared_k(\hat{p})}{pred^i_k(\hat{p})}. \tag{3.2}$$

[1] Although this test can be applied formally to positive semidefinite matrices also, the meaning of V_k is ambiguous in this case.

Values close to one denote a "good" model. This test seems very natural given the success of trust region methods that use this ratio to control the trust radius. It is not limited to positive definite quadratics but can be applied to any general model for which a predicted reduction can be computed. Furthermore, it can be used both for initial model selection and for model switching. The test step $\hat{p}$ for selection of the trial model is typically the iterate p_k just computed. For model switching, $\hat{p}$ is typically the most recently computed trial step. A function evaluation is required whenever this test is used, but in many cases this function evaluation would have been required anyway.

Several implementations using this test are investigated by Al-Baali, Fletcher [1985]. Their algorithms use the Gauss-Newton model until ρ for this model becomes unacceptable ($\rho < 0.05$) and then switch to a BFGS model. Nazareth [1980,1983] uses this test to control an algorithm that superficially looks different from A.3.3. Instead of choosing between a finite set of models, Nazareth uses a model Hessian which is a convex combination of two other model Hessians, say

$$B_k \equiv \alpha_k \; B_k^1 + (\; 1 - \alpha_k \;) \; B_k^2 \tag{3.3}$$

for some $\alpha_k \in [\; 0 \; , \; 1 \;]$. For the nonlinear least squares problem, if one model is the Gauss-Newton model and the other is a structured secant method, this is equivalent to choosing a scalar multiple of our secant approximation to the second order term $S(x_k)$ (see Example 1.2). Although the algorithm draws from an infinite set of possible models rather than a finite set, only one model is chosen at each iteration so that the algorithm is no different in principle from other model selection algorithms.

The balance between the two original models is initially set to 0.5 and is adjusted by the following procedure.

Procedure A.3.4 : Nazareth's Method for Choosing α_k.

a) Compute $\rho_{k-1}^1(p_{k-1})$ and $\rho_{k-1}^2(p_{k-1})$.

b) If $\rho_{k-1}^1(p_{k-1})$ is closer than $\rho_{k-1}^2(p_{k-1})$ to 1,

```
    then bias toward model one :
        If $\alpha_{k-1}=0$, set $\alpha_k=.05$.
        If $\alpha_{k-1}\in(0,\frac{1}{3}]$, set $\alpha_k=2\alpha_{k-1}$.
        If $\alpha_{k-1}\in(\frac{1}{3},.95)$, set $\alpha_k=(1+\alpha_{k-1})/2$.
        If $\alpha_{k-1}\in[.95,1]$, set $\alpha_k=1$.

    Else bias toward model two :
        If $\alpha_{k-1}=1$, set $\alpha_k=.95$.
        If $\alpha_{k-1}\in[\frac{2}{3},1)$, set $\alpha_k=2\alpha_{k-1}-1$.
        If $\alpha_{k-1}\in(.05,\frac{2}{3})$, set $\alpha_k=\alpha_{k-1}/2$.
        If $\alpha_{k-1}\in[0,.05]$, set $\alpha_k=0$.
```

c) Proceed

Note that rather than using the ρ test on the model actually used, this method applies it to the two "base" models B_k^1 and B_k^2 and then generates the model B_k to be used. More complicated procedures are obviously possible, but since only one model is finally chosen at each iteration, any such technique would still be a model selection algorithm.

A third useful test is similar to the ρ test in that it compares predicted reduction to actual reduction for specific test steps. As in the ρ test, $\hat{p}$ is typically p_{k-1} or p_k^i. Given a test step $\hat{p}$, an **e - test** computes the **observed error :**

$$e_k^i(\hat{p}\ ) \equiv |\ ared_k(\hat{p}) - pred_k^i(\hat{p})\ | \ . \qquad (3.4)$$

Values close to zero denote a "good" model. This test is used in NL2SOL and is the preferred test in the model switching algorithms of Chapter 5. Its primary difference from the ρ test is that it lacks a "*pred*" term in the denominator. Leaving out this term eliminates certain sensitivity problems associated with small predicted reductions.

3.3.1.3. Some Properties of these Tests

As previously stated, the really interesting question about these tests is whether any can distinguish between a sequence of model Hessians which is uniformly bounded and a sequence which is unbounded. We now demonstrate that none of the methods of initially choosing a model discussed in the literature are capable of always rejecting an arbitrarily bad model.

Proposition 3.1 : *There exist two sequences of model Hessians, denoted $\{B_k^1\}$ and $\{B_k^2\}$, with $||B_k^1|| \leq \beta_3$ for some $\beta_3 > 0$, and with $\{||B_k^2||\}$ unbounded for which the variational test can select a model from the second group at every iterate.*

Proof : We prove this proposition by construction. Al-Baali, Fletcher [1985] give the following representation for $V_k(B)$:

$$V_k(B) = \left[\left[\frac{y_k^t B^{-1} y_k}{y_k^t p_k} \right]^2 - 2 \frac{y_k^t p_k}{p_k^t B p_k} + 1 \right]^{\frac{1}{2}} \tag{3.5}$$

where the weight matrix U satisfies the secant equation

$$U\, p_k = y_k \tag{3.6}$$

for some nonzero $y_k \in \mathbb{R}^n$.[2] Suppose that $n \geq 3$, and define a set of vectors $\{w_k\}$ in $\mathbb{R}^n$ satisfying

$$\begin{aligned} w_k^t p_k &= 0 \\ w_k^t y_k &= 0 \end{aligned} \tag{3.7}$$

and

$$\| w_k \| = 1 . \tag{3.8}$$

Consider two sequences of model Hessians defined by

$$B_k^1 = I \tag{3.9}$$

and

$$B_k^2 = I + k!\; w_k w_k^t \ . \tag{3.10}$$

Now, by the Sherman-Morrison-Woodbury[3] formula, $(B_k^2)^{-1}$ exists and has value

$$(B_k^2)^{-1} = I - \frac{k!}{1+k!} w_k w_k^t \ . \tag{3.11}$$

[2] The standard definition for y_k is $g_{k+1} - g_k$, so that U interpolates the observed change in g, but other variations are typically used for NLS to take advantage of problem structure.

[3] See, for example, Dennis and Schnabel [1983], Lemma 8.3.1.

Then

$$y_k^t(B_k^2)^{-1}y_k = y_k^t I y_k - \frac{k!}{1+k!}\left[w_k^t y_k\right]^2 \\ = y_k^t\,(B_k^1)^{-1}y_k \quad . \tag{3.12}$$

Moreover,

$$p_k^t B_k^2 p_k = p_k^t I p_k + k!(p_k^t w_k)^2 \\ = p_k^t B_k^1 p_k \ . \tag{3.13}$$

Therefore combining (3.5), (3.12), and (3.13) gives

$$V_k(B_k^1) = V_k(B_k^2) \ . \tag{3.14}$$

Thus the variational test cannot distinguish between the models, which allows model two to be selected. □

Proposition 3.2 : *There exist two sequences of model Hessians, denoted* $\{B_k^1\}$ *and* $\{B_k^2\}$, *with* $\| B_k^1 \| \le \beta_3$ *for some* $\beta_3 > 0$ *and with* $\{\| B_k^2 \|\}$ *unbounded for which both the* ρ *test and the e-test can generate a sequence of models with*

$$\limsup_{k \to \infty} \| B_k \| = \infty \ .$$

Proof : We prove this proposition by construction. Let $f(x) = \frac{1}{2}x^t x$. Assume first that the model to be used in step $k+1$ is chosen *after* the models have been updated. Consider two sequences of model Hessians defined by

$$B_k^1 = 2I \tag{3.15}$$

and

$$B_k^2 = k!I - (k!-1)\frac{p_{k-1}p_{k-1}^t}{p_{k-1}^t p_{k-1}} \ . \tag{3.16}$$

Then

$$\begin{aligned} pred_k^1(p_{k-1}) &= -g_k^t p_{k-1} - \frac{1}{2} p_{k-1}^t \, (2I) p_{k-1} \\ &= ared_k(p_{k-1}) - \frac{1}{2} \| p_{k-1} \|_2^2 \quad , \end{aligned} \tag{3.17}$$

but

$$\begin{aligned} pred_k^2(p_{k-1}) &= -g_k^t p_{k-1} - \frac{1}{2} p_{k-1}^t \left[k!I - (k!-1) \frac{p_{k-1} p_{k-1}^t}{p_{k-1}^t p_{k-1}} \right] p_{k-1} \\ &= ared_k \, (p_{k-1}) \, . \end{aligned} \tag{3.18}$$

Therefore both the ρ and e-tests would choose model Hessian $B_k = B_k^2$ over B_k^1 even though B_k^2 is arbitrarily bad in every subspace perpendicular to p_{k-1}.

If the choice is made before the models have been updated, then consider the two sequences of model Hessians

$$\begin{aligned} B_k^1 &= 2I \\ B_k^2 &= (1+(-1)^k) k!I + I \end{aligned} \tag{3.19}$$

Then

$$pred_k^2 \, (p_k) = \begin{cases} ared_k \, (p_k) & k \ odd \\ ared_k \, (p_k) + k! \, \| p_k \|_2^2 & k \ even \end{cases} \tag{3.20}$$

As before, both the ρ test and the e-test would select model two at the end of each odd numbered iteration, and thus the algorithm would use $B_k = B_k^2 = 2k!I + I$ throughout each even numbered iteration. □

Proposition 3.3 : *There exist two sequences of model Hessians, denoted* $\{B_k^1\}$ *and* $\{B_k^2\}$, *with* $\| B_k^1 \| \leq \beta_3$ *for some* $\beta_3 > 0$, *and with* $\{ \| B_k^2 \| \}$ *unbounded for* *which the* ρ *test combined with interpolation between models as used in Nazareth can generate a sequence of models* $\{B_k\}$ *as in equation (3.3) with*

$$\limsup_{k \to \infty} \| B_k \| = \infty \ . \tag{3.21}$$

Proof : We prove this proposition by construction. Let

$$f(x) = \frac{1}{2} x^t x \ . \tag{3.22}$$

Consider two sequences of model Hessians defined by

$$B_k^1 = I\,(2+(-1)^k) \tag{3.23}$$

and

$$B_k^2 = I\,(1+(1-(-1)^k)\,k!) \ . \tag{3.24}$$

Now, at any iteration where k is odd, we have

$$\rho_k^1(p) = 1 + \frac{1}{2} \frac{\|p\|^2}{-g_k^t p - p^t p} \tag{3.25}$$

and

$$\rho_k^2(p) = 1 \ , \tag{3.26}$$

so that α_{k+1} is biased toward the second model. At any iteration where k is even, we have

$$\rho_k^1(p) = 1 \tag{3.27}$$

$$\rho_k^2(p) = 1 + (k! - \frac{1}{2}) \left[\frac{\|p\|^2}{-g_k^t p - k!\,\|p\|^2} \right] \ , \tag{3.28}$$

so that α_{k+1} is biased toward the first model. Thus, since α_1 is initialized to $\frac{1}{2}$, we have that α alternates between $\frac{1}{2}$ and $\frac{3}{4}$, and

$$B_k = \begin{cases} \frac{5}{4} I & k \ odd \\ (k! + \frac{1}{2}) I & k \ even \end{cases} \tag{3.29}$$

Thus,

$$\limsup_{k \to \infty} \| B_k \| = \infty \ .$$

□

After considering these examples, it seems doubtful that any model *selection* algorithm that selects a model solely on its performance at the last step can have the properties we want. Since we are not allowed to switch models after the start of an iteration, the scheme used to initially choose this model *must* select the correct one. We have seen how easy it is to produce pathological special cases to fool the existing tests.[4] However, model *switching* algorithms *can* be constructed which do not share these (potential) shortcomings. With this in mind, model *switching* algorithms are to be recommended over model *selection* algorithms.

3.3.2. Model Switching Algorithms

Chapters 4 to 7 deal with this type of algorithm in detail. Since switching models is allowed after the current iteration has been started, we would expect that the harsh

[4] It could be argued, with some justification, that since the counterexamples in Propositions 3.1, 3.2, and 3.3 are for a set of "arbitrarily bad" models rather than a set of "arbitrarily bad models generated by some method of interest," model selection logic might still link with the model generation scheme in a manner guaranteeing global convergence anyway. A counterexample to this possibility cannot be given for the simple reason that, for methods of interest such as the BFGS, boundedness of the Hessian approximations is still an open question. In lieu of more knowledge, we must assume that if there is the possibility of one of the secant methods generating a sequence Hessians of unbounded norm, then there is the possibility that these sequences may have the same structure as our counterexamples.

requirements on correctness of the initial trial model can be relaxed. This does prove to be the case. We show that when properly constructed, the model switching portion of these algorithms does play a nontrivial role in assuring global convergence. A wide range of methods for model switching and trust region modification are allowed. Furthermore, three major versions of the basic algorithm are presented which explore the tradeoffs between restrictions on the models and restrictions on the details of the algorithm. Type one algorithms require all models to be quadratic with uniformly bounded curvature but have essentially no restrictions on the details of the switching system. Type two algorithms require at least one model to be quadratic with uniformly bounded curvature. Each other model is required to generate trial steps satisfying a ''uniform predicted decrease'' condition. The switching system is slightly more restricted, in that it must limit the circumstances under which the trust radius can be reduced. Several schemes are defined which do this. Type three algorithms only require one model to be quadratic with uniformly bounded curvature.

3.3.3. Concurrent Model Algorithms

This category of algorithms is a ''catch all'' for methods that are more complicated than selection or switching procedures. At present, the extra complications involved in most of these algorithms appear unjustified, and too many disparate formulations exist to allow a general framework. However, the following examples are included for completeness. The first one is trivial but the second is quite intriguing.

Many -- or even most -- implementations of optimization algorithms contain logic which is implicitly or explicitly equivalent to using a concurrent scheme.

Example 3.1 : Any algorithm which makes some decision based on whether the norm of some model Hessian $\hat{B}$ is greater than a constant β_1 is equivalent to an algorithm making the same decision based on whether the norm of $\hat{B}$ is greater than the norm of $\bar{B}$ where $\bar{B} = \beta_1 I$. Even the Armijo-Goldstein condition used for line searches can be considered a concurrent model algorithm, where the ''alternate'' model Hessian (used only for testing trial steps and not computing them) is $\bar{B} = 0$.

Concurrent model algorithms can also be fundamentally different from anything previously considered.

Example 3.2 : Rather than choosing one model or the other to compute a trial step, this step can be computed by *reconciling* the models. The first model can be used to define a region $\Omega^1 = \{ p : p$ satisfies (2.22) for $B_k \equiv B_k^1 \}$. A step p_k can then be computed to maximize the predicted reduction of the second model over the intersection of Ω^1 and the trust region. To further complicate the algorithm, one could also allow the option of computing a step to maximize the predicted reduction of the first model over the intersection of Ω^2 and the trust region (where Ω^2 is defined in the obvious fashion). Although this is a fascinating idea, global convergence can be guaranteed by the simpler model switching algorithms. Furthermore, an example in Chapter 7 casts doubt on the local convergence properties of this algorithm, and the existing techniques for solving the modified subproblem are not as numerically efficient as the more mature methods in existence for solving problem TRS. Although some versions of multi-model algorithms of this type may yet prove workable, our current belief is that none will prove competitive with the other algorithms presented for unconstrained optimization. More about using the modified subproblem above in constrained optimization can be found in Celis, Dennis, and Tapia [1984] . This approach is also similar to ideas from multi-objective optimization (see, for example, Woods [1985], Chankong and Haimes [1983], or Ignizio [1982]).

3.4. Further Useful Concepts

Additional concepts -- which are useful in actually writing an implementation or helpful in attaining a complete understanding of multi-model methods, but which are of small importance to the theory presented in later sections -- are presented in Appendix B.

CHAPTER 4

Safe Algorithms using Maximal Restrictions on the Models

4.1. Introduction

This chapter analyses the global convergence properties of model switching algorithms when all of the models are quadratic and when we are free to assume as many restrictions on the models involved as we need. We show that if there exists a uniform upper bound on the norm of each model Hessian available to the system, then essentially *any* switching system gives first order stationary point convergence. This generalizes the first order results of SSB [1985] to the multiple model case.

4.2. Preliminary Definitions and Results

The notation in this chapter is largely an extension of that described in Chapter 2 for single model trust region algorithms. Superscripts are included to distinguish among models. For definiteness, we now present (or review) all of the notation necessary for this chapter. The reader is again encouraged to make use of the glossary in Appendix A.

4.2.1. Local Models

Assume that at any iteration we have a finite set of symmetric matrices $\{B_k^i\}$, $i=1,\ldots,N_k$ representing approximations to the Hessian of f at x_k. This gives us N_k **quadratic models in standard form**

$$pred_k^i(p) = -g_k^t p - \frac{1}{2} p^t B_k^i p \tag{4.1}$$

for use in approximating the function in local coordinates.

For quadratics, most algorithms guarantee that any step p_k^i computed as an approximate solution to problem TRS satisfies one of the following two conditions. Let $c_1 > 0$ and $c_2 > 0$ be two constants which are independent of k. Steps that satisfy

$$pred_k^i(p_k^i) \geq c_1 \max_{\substack{\alpha \\ s/t \, \|\alpha g_k\| \leq \Delta_k}} pred_k^i(\alpha g_k) \tag{4.2}$$

are called **FCD** (Fraction of Cauchy Decrease) steps. Steps of this type are generated by dogleg and double dogleg procedures, generalized doglegs, and restricted subspace methods as defined in Chapter 2. Steps that satisfy

$$pred_k^i(p_k^i) \geq c_2 \max_{\substack{p \\ s/t \, \|p\| \leq \Delta_k}} pred_k^i(p) \tag{4.3}$$

are generated by **OLC** (Optimal Locally Constrained) procedures. Such steps can be computed, for example, as in Moré and Sorensen [1983].[1] The matrix B_k^i need not be nonsingular or even positive definite. An OLC step also satisfies (4.2).

[1] If g_k is zero at any iteration, some of the published techniques may fail to guarantee (4.3). However, as we are only concerned with FOSPC, we consider $g_k = 0$ at any iteration to constitute successful completion of the algorithm.

The following lemma is well known.

Lemma 4.1 : *If p_k^i is an FCD step with constant $c_1>0$ or an OLC step with constant $c_2>0$ for a quadratic model in standard form with model Hessian B_k^i, then there exists $c_3 \in (0,1)$ independent of k such that*

$$pred_k^i\ (p_k^i) \geq c_3\ \|g_k\|\ \min\left\{ \Delta_k\ ,\ \frac{\|g_k\|}{\|B_k^i\|} \right\}\quad . \tag{4.4}$$

More specifically, every $c_3 \leq \frac{1}{2} c_1$ satisfies (4.4) for an FCD step, and every $c_3 \leq \frac{1}{2} c_2$ satisfies (4.4) for an OLC step.

Proof of this lemma can be found, for example, in Powell [1975]. This is also a special case of Theorem 6.4.

If $\|B_k^i\|$ is bounded as $k \to \infty$, the sequence of models is said to have **uniformly bounded curvature.** By Lemma 4.1, if each p_k^i is an FCD or OLC step and $\|B_k^i\| \leq \beta_2$, then

$$pred_k^i\ (p_k^i) \geq c_3\ \|\ g_k\ \|\ \min\left\{ \Delta_k\ ,\ \frac{\|g_k\|}{\beta_2} \right\}\quad . \tag{4.5}$$

For a given positive definite model Hessian B_k^i and a sufficiently large trust region, most implementations generate a step to the global minimizer of the model. A **quasi-Newton step** is a step p_k^i that satisfies

$$B_k^i\ p_k^i = -\ g_k\quad . \tag{4.6a}$$

For large sparse systems, implementations often do not require exact computation of this step but instead impose the following condition on how accurately the next iterate should be calculated. A step p_k^i is said to satisfy the **inexact quasi-Newton condition** if

$$\| B_k^i p_k^i + g_k \| \le \zeta_k \| g_k \| \tag{4.6b}$$

where $0 \le \zeta_k < 1$ and $\lim_{k \to \infty} \zeta_k = 0$. More about this condition can be found in Steihaug [1980]. Any quasi-Newton step automatically satisfies the inexact quasi-Newton condition.

Most versions of trust region algorithms enforce one of the following sets of conditions relating the size of a computed step to Δ_k. When a restricted subspace or a dogleg type scheme is used to approximately solve the trust region subproblem, a step satisfies either

$$\| p_k^i \| = \Delta_k \tag{4.7a}$$

or

$$\text{((4.6a) or (4.6b)) and } \| p_k^i \| < \Delta_k \quad . \tag{4.7b}$$

That is, each step is either on the boundary of the trust region, or it is a quasi-Newton (or inexact quasi-Newton) step. An OLC step satisfies either

$$(1 - \sigma_1) \Delta_k \le \| p_k^i \| \le (1 + \sigma_1) \Delta_k \tag{4.8a}$$

or

$$\text{((4.6a) or (4.6b)) and } \| p_k^i \| < (1 - \sigma_1) \Delta_k \quad . \tag{4.8b}$$

In either event, and for all other reasonable implementations, a trust region algorithm enforces

$$\|p_k^i\| \leq (1 + \sigma_1) \Delta_k \tag{4.9}$$

for some constant $\sigma_1 \in [0 , 1)$.

Each of our models and associated procedures for computing steps may differ considerably in the conditions they guarantee. In order to avoid overspecifying our algorithm at this point we introduce the following notation.

Definition 4.1 : We use the notational shorthand $p_k^i \in \mathrm{A}(i,\Delta_k)$ to denote any approximation to the solution of the trust region subproblem that is **acceptable** with respect to the model with index i and the trust radius Δ_k. If the index of the model is clear from context, we may also use the notation $p_k \in \mathrm{A}(\Delta_k)$.

The precise meaning of "acceptable" will be specified separately for each class of algorithms.

4.2.2. The Standard Assumptions on the Function

The results and definitions of the previous section applied strictly to the *models* being used. Other than the occasional use of g_k, no reference has been made to the actual function being approximated. In this section we present notation related to the function itself. The expression

$$ared_k(p) = f(x_k) - f(x_k + p) \tag{4.10}$$

again denotes the reduction in f caused by a step p. For the $i^{\underline{\text{th}}}$ model, the ratio of actual reduction to predicted reduction is defined as

$$\rho_k^i(p) \equiv \frac{ared_k(p)}{pred_k^i(p)} \quad . \tag{4.11}$$

Next, recall that $L(f, x_1)$ denotes of the level set of f at x_1.

Definition 4.2 : For given $x_1 \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$, the following conditions are referred to as the **standard assumptions on the function.**

(SA1) f is twice continuously differentiable on an open convex set containing $L(f, x_1)$. This open set is denoted $\tilde{L}(f, x_1)$.

(SA2) f is bounded below on $L(f, x_1)$.

(SA3) $\exists\ \beta_1 > 0$ such that for all $x \in \tilde{L}(f, x_1)$, the Hessian of f satisfies $\| H(x) \| \leq \beta_1$.

4.2.3. Some Useful Results

The next lemma is invaluable in establishing the existence of acceptable steps in specific stages of an algorithm. Let σ_1, c_3 be defined so that $\sigma_1 \in [0, 1)$, $c_3 \in (0,1)$, and consider the following conditions.

(a) $\| p \| \leq (1 + \sigma_1) \Delta_k$.

(b) $pred_k^i(p) \geq c_3 \| g_k \| \min \{ \Delta_k , \dfrac{\| g_k \|}{\| B_k^i \|} \}$.

Lemma 4.2 : *For a given $x_1 \in \mathbb{R}^n$, let $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$ satisfy the standard assumptions and suppose that for some k, the vector x_k satisfies $x_k \in L(f, x_1)$ and $g(x_k) \neq 0$.*

Then there exists $\delta_k^0 > 0$ *such that* $\Delta_k \in (0, \delta_k^0]$ *and condition (a) imply*

$$x_k + p \in \tilde{L}(f, x_1) \quad . \tag{4.12}$$

Furthermore, let B_k^i *be the model Hessian of a quadratic model in standard form. For any* $\eta \in (0,1)$, *let*

$$\delta_k^i \equiv \frac{2(1-\eta)\, c_3 \, \| g_k \|}{(1+\sigma_1)^2 (\beta_1 + \| B_k^i \|)} \tag{4.13}$$

If $\Delta_k \in (0, \delta_k^i]$ *and* $x_k + p \in \tilde{L}(f, x_1)$, *then (a) and (b) imply*

$$2 - \eta \geq \rho_k^i(p) \geq \eta \quad . \tag{4.14}$$

Proof : From the hypotheses and the definition of (SA1), x_k is in the open set $\tilde{L}(f, x_1)$, which immediately implies the existence of a positive δ_k^0 for which $\Delta_k \leq \delta_k^0$ and (a) imply (4.12).

Next, from Lemma 2.6 and assumptions (SA1), (SA3), (4.12), (a), and (b) we have

$$| 1-\rho_k^i(p) | = \left| \frac{ared_k(p) - pred_k^i(p)}{pred_k^i(p)} \right|$$

$$\leq \left| \frac{\int_0^1 (p)^t (B_k^i - H(x_k + \xi p))(p) (1 - \xi) \, d\xi}{c_3 \, \| g_k \| \, \min\left\{ \Delta_k \, , \, \frac{\| g_k \|}{\| B_k^i \|} \right\}} \right| \tag{4.15}$$

$$\leq \frac{\| p \|^2 (\| B_k^i \| + \beta_1)}{2 c_3 \, \| g_k \| \, \min\left\{ \Delta_k \, , \, \frac{\| g_k \|}{\| B_k^i \|} \right\}} \; .$$

From (a) we then have

$$| 1-\rho_k^i(p) | \leq \frac{(\| B_k^i \| + \beta_1) (1+\sigma_1)^2 \Delta_k^2}{2 c_3 \, \| g_k \| \, \min\left\{ \Delta_k \, , \, \frac{\| g_k \|}{\| B_k^i \|} \right\}} \; . \tag{4.16}$$

Since $c_3 < 1$ and $\Delta_k \leq \delta_k^i$, (4.13) implies

$$\Delta_k \leq \frac{\| g_k \|}{\| B_k^i \|} \; . \tag{4.17}$$

From (4.13), (4.16) and (4.17) we obtain

$$\begin{aligned} | 1-\rho_k^i(p) | &\leq \frac{(\| B_k^i \| + \beta_1) (1+\sigma_1)^2 (\Delta_k)^2}{2 c_3 \, \| g_k \| \, \Delta_k} \\ &\leq \frac{(\| B_k^i \| + \beta_1) \, (1+\sigma_1)^2}{2 c_3 \, \| g_k \|} \, \delta_k^i \qquad (4.18) \\ &\leq (1-\eta) \end{aligned}$$

which establishes (4.14). □

Consider a sequence $\{x_k\} \in \mathbb{R}^n$. For every $m > 0$, define Ω_m to be the set

$$\Omega_m \equiv \{ \, x : \| x - x_m \| < \frac{\| g_m \|}{2\beta_1} \text{ and } x \in L(f, x_1) \, \} \ . \tag{4.19}$$

Lemma 4.3 : *If f satisfies the standard assumptions, then for any $x \in \Omega_m$,*

$$\frac{3}{2} \| g_m \| \geq \| g(x) \| \geq \frac{1}{2} \| g_m \| \ . \tag{4.20}$$

Proof : If $x_m \notin L(f, x_1)$, then Ω_m is empty, so that (4.20) is immediately established. Consider any m with $x_m \in L(f, x_1)$. For any $x \in L(f, x_1)$, the standard assumptions and Lemma 2.5 imply $\| g(x) - g_m \| \leq \beta_1 \| x - x_m \|$. By the triangle inequality,

$$\| g_m \| + \| g(x) - g_m \| \geq \| g(x) \| \geq \| g_m \| - \| g(x) - g_m \| \tag{4.21}$$

so if $\| x - x_m \| < \| g_m \| / 2\beta_1$, then

$$\frac{3}{2} \| g_m \| \geq \| g(x) \| \geq \frac{1}{2} \| g_m \| \tag{4.22}$$

which establishes the result. □

Next, consider a sequence of positive numbers $\{\Delta_k\}$, and let p_k be defined by $p_k = x_{k+1} - x_k$. Furthermore, consider the following conditions.

(i) There exists $\eta_1, c_3 \in (0,1)$ and $\beta_2 > 0$ such that each p_k satisfies the **uniform decrease** condition :

$$ared_k(p_k) \geq \eta_1 c_3 \| g_k \| \min \left\{ \Delta_k \ , \ \frac{\| g_k \|}{\beta_2} \right\} . \tag{4.23}$$

(ii) There exists $\sigma_1 \in [0,1)$ such that each p_k satisfies $\| p_k \| \le (1+\sigma_1)\Delta_k$.

The next theorem establishes FOSPC for $\{x_k\}$ satisfying (i) and (ii), provided the sequence $\{x_k\}$ does not remain within a certain ball about any given iterate x_m. Proof of this result can essentially be found in SSB [1985], although the theorem was not directly stated therein.

Theorem 4.4 : *Let f satisfy the standard assumptions, and assume that (i) and (ii) are true. If for every $m > 0$, either*

(a) $g_m = 0$, or

(b) $\exists \bar{m} > m$ such that $\| x_{\bar{m}} - x_m \| \ge \dfrac{\| g_m \|}{2\beta_1}$,

then $g_k \to 0$ (FOSPC).

Proof : First notice that (i) implies $ared_k(p_k) \ge 0$, so that $\{x_k\}$ satisfies the descent condition. Hence $x_k \in L(f, x_1)$ for all $k > 0$ and $f(x_{k+1}) \le f(x_k)$ for all $k > 0$. Now, consider any m with $\| g_m \| \ne 0$. By assumption, $\exists \bar{m} > m$ such that $x_{\bar{m}} \notin \Omega_m$. Let $l+1$ denote the first such index after m with x_{l+1} not in Ω_m. Then

$$\begin{aligned} \frac{\| g_m \|}{2\beta_1} &\le \| x_{l+1} - x_m \| \le \sum_{k=m}^{l} \| p_k \| \\ &\le (1+\sigma_1) \sum_{k=m}^{l} \Delta_k \ , \end{aligned} \tag{4.24}$$

so that Lemma 4.3 implies

$$\begin{aligned} f(x_m) - f(x_{l+1}) &= \sum_{k=m}^{l} \Big[f(x_k) - f(x_{k+1}) \Big] \\ &\geq \sum_{k=m}^{l} \eta_1 \; c_3 \, \| g_k \| \; \min \left\{ \Delta_k \; , \; \frac{\| g_k \|}{\beta_2} \right\} \\ &\geq \sum_{k=m}^{l} \frac{1}{2} \eta_1 \, c_3 \; \| g_m \| \; \min \left\{ \Delta_k \; , \; \frac{\| g_m \|}{2\beta_2} \right\} \; . \end{aligned} \tag{4.25}$$

Now, if $\Delta_k \leq \| g_m \| / 2\beta_2$ for $m \leq k \leq l$, (4.24) and (4.25) imply

$$\begin{aligned} f(x_m) - f(x_{l+1}) &\geq \frac{1}{2} \, \eta_1 \, c_3 \; \| g_m \| \; \sum_{k=m}^{l} \Delta_k \\ &\geq \frac{1}{2} \eta_1 \, c_3 \; \| g_m \| \; \frac{\| g_m \|}{2 \beta_1 (1 + \sigma_1)} \quad . \end{aligned} \tag{4.26}$$

Otherwise,

$$f(x_m) - f(x_{l+1}) \geq \; \eta_1 \; c_3 \; \frac{\| g_m \|^2}{4 \beta_2} \quad . \tag{4.27}$$

We can then write

$$f(x_m) - f(x_{l+1}) \geq \frac{\eta_1 \; c_3}{4} \, \| g_m \|^2 \; \min \left\{ \frac{1}{\beta_1 (1 + \sigma_1)} \; , \; \frac{1}{\beta_2} \right\} \; . \tag{4.28}$$

Now, since f is bounded below and $\{ f(x_k) \}$ is nonincreasing, $\{ f(x_k) \}$ converges to some limit, say $f(x_k) \rightarrow f^*$. Thus for any m, either $g_m = 0$ or

$$\| g_m \|^2 \leq \left[\frac{\eta_1 \; c_3}{4} \; \min \left\{ \frac{1}{\beta_1 (1 + \sigma_1)} \; , \; \frac{1}{\beta_2} \right\} \right]^{-1} \; (\, f(x_m) - f^* \,) \; . \tag{4.29}$$

Therefore $g_k \rightarrow 0$, which completes the proof. □

4.3. Introductory Examples

Before considering a general definition of the algorithm, let us write two simple examples. We follow the ''minimal version'' trust region Algorithm A.2.1, but construct it for Example 1.2 of Chapter 1. Recall that in this example we sketched a method (Algorithm A.1.5) that makes use of a constant matrix for the model Hessian when computation of the exact Hessian is too expensive to be done frequently.

Algorithm A.4.1 : Constant Matrix versus an Exact Hessian

```
0) Let $x_1 \in \mathbb{R}^n, \Delta_1 > 0$ and a symmetric $B \in \mathbb{R}^{n \times n}$ be given.
1) Repeat until convergence :
    2) Set $B_k = B$.
    3) Compute $p_k \in A(\Delta_k)$.
    4) If $ared_k(p_k) < 0$, then set $B_k = H(x_k)$.
    5) If $\rho_k(p_k) < \frac{1}{100}$, then
        If $0 < ||p_k|| \le \Delta_k$ then set $\Delta_k = ||p_k||/4$,
            Else set $\Delta_k = \Delta_k/4$.
        Return to 3).
    6) Set $x_{k+1} = x_k + p_k$ , $\Delta_{k+1} = \Delta_k$.
    7) If $\rho_k(p_k) \ge \frac{1}{2}$, then set $\Delta_{k+1} = 2\Delta_k$ .
8) End.
```

Model switching is done in step 4) by the simple expedient of evaluating $H(x_k)$ whenever a test step generated by the constant model yields a function increase. This represents just one possible way of deciding whether to compute the full Hessian. In the next algorithm, an addition is made to this test so that it is only performed when the trust radius is less than a fixed tolerance. A slightly different version of A.1.5 is used, so that the most recently computed Hessian is always used for B_k.

Algorithm A.4.2 : Exact Hessian evaluated intermittently

```
0) Let $x_1 \in \mathbb{R}^n, \Delta_1 > 0$ and a symmetric $B_1 \in \mathbb{R}^{n \times n}$ be given.

1) Repeat until convergence :
    2) Compute $p_k \in A(\Delta_k)$.
    3) If $\Delta_k \le 10^{-6}$ and $ared_k(p_k) < 0$, then set $B_k = H(x_k)$.
    4) If $\rho_k(p_k) < \frac{1}{100}$, then
        If $0 < \|p_k\| \le \Delta_k$ then set $\Delta_k = \|p_k\|/4$,
            Else set $\Delta_k = \Delta_k/4$.
        Return to 2).
    5) Set $x_{k+1} = x_k + p_k$ , $\Delta_{k+1} = \Delta_k$.
    6) If $\rho_k(p_k) \ge \frac{1}{2}$, then set $\Delta_{k+1} = 2\ \Delta_k$ .
7) End.
```

This implementation refreshes the constant matrix whenever the trust radius drops beneath an arbitrary lower bound and a computed step does not decrease the function value. Any other way of model switching or selection could also be implemented without affecting the theory of this chapter, such as refreshing every mth iteration, refreshing when $H(x_k)$ is positive definite, or refreshing when $\|g_k\|$ is smaller than a certain value.

These two sample algorithms are probably only useful for unusual classes of problems involving special structure or extremely expensive Hessian or gradient evaluations. In actual implementation, the precise application and any knowledge about the relative computational costs should suggest a particular technique for choosing a model. If not, we suggest refreshing B_k whenever one of the following holds.

(1) The trust radius drops to less than a fraction of the largest trust radius used since the last instance of refreshing B_k.

(2) An iterate x_k seems to be near enough to the solution so that Newton's method is quadratically convergent without step size control.

One should also include a more sophisticated trust region update strategy. Such a strategy should include the capability of rapidly increasing Λ_k after refreshing B_k.

4.4. The Basic Algorithm Framework

We first define a general framework for a model switching algorithm. Most details are initially unspecified to emphasize the points at which choices can be made in designing an implementation. We document exactly what choices are allowed in Algorithms AS.4.4 and AS.4.6.

Algorithm A.4.3 : General Model Switching

```
0) Let $k=1$, $x_1 \in \mathbb{R}^n$ , a trial trust radius $\bar{\Delta}_1>0$ , a set of models indexed by
   $i=1,2,\ldots,N_1$ , and an initial trial model with index $\bar{i}$ all be given.

1) Repeat until convergence :

    2) Compute an acceptable step $p_k$ as follows :

      Either
        2.1) Find a trust radius $\Delta_k > \bar{\Delta}_k$, a model with index $i$, and a step $p_k$
               for which $p_k = p_k^i \in A(i,\Delta_k)$,
      Or
        2.2) Compute a trial step $p_k^{\bar{i}} \in A(\bar{i},\bar{\Delta}_k)$. Then either
             (a) Accept $p_k = p_k^{\bar{i}}$, $\Delta_k = \bar{\Delta}_k$, and go to 3) , or
             (b) Proceed to 2.3) for trust radius reduction , or
             (c) Choose a different trial model and repeat 2.2).

        2.3) Find a trust radius $\Delta_k < \bar{\Delta}_k$, a model with index $i$, and a step $p_k$
               for which $p_k = p_k^i \in A(i,\Delta_k)$.

    3) Choose a trial radius $\bar{\Delta}_{k+1}$ for the next iteration.

    4) Perform any calculations required to make the models
           { $pred_{k+1}^j$ } , $j=1,2,\ldots,N_{k+1}$ available at the next iteration.
       Select an initial trial model with index $\bar{i}$ for the next iteration.
       Set $x_{k+1}=x_k+p_k$ , $f_{k+1}=f_k$ , $g_{k+1}=g(x_{k+1})$ , $k=k+1$.

5) End.
```

In this most basic form of the algorithm, we have not specified several things.

(1) How to choose among the options in step 2).

(2) How to implement whichever one we choose.

(3) How to choose a trial radius for the next iteration.

The allowable methods for doing these steps differ depending on what is known about the models. For standard quadratic models with uniformly bounded curvature, Section 4.5 shows the choices allowed in the logic for deciding which option to execute in step 2), and Section 4.6 presents the allowable logic for step 3).

Typical values for the parameters used throughout the rest of this thesis can be found in Appendix A.

4.5. Computation of an Acceptable Step

Step 2) in A.4.3 is usually referred to as the ``Internal Loop'' section of the algorithm. The bulk of model switching (as opposed to model selection) logic must lie in this section. Our theory does not put any restrictions on the initial choice of a model, nor does it require 2.2) to be tried before 2.1). The following specifications are sufficient to define the range of allowable options in step 2).

Algorithm Specification AS.4.4 : Permissible Choices in Step 2) at the k$^{\underline{th}}$ Iteration

Let $0<\eta_1<\eta_2<1$, a trial trust radius $\bar{\Delta}_k>0$, and a set of N_k models be given.

2.1) We are *allowed* to choose $\Delta_k > \bar{\Delta}_k$ and find p_k^i for some model with index i satisfying both $p_k = p_k^i \in A(i,\Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$ *whenever* such a Δ_k , model , and corresponding step can be found.

2.2) We are *allowed* to accept $\Delta_k = \bar{\Delta}_k$ and find p_k^i for some model with index i satisfying both $p_k = p_k^i \in A(i,\Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$ *whenever* such a Δ_k , model , and corresponding step can be found.

If *at least one* model with index j and associated trial step $p_k^j \in A(j,\bar{\Delta}_k)$ computed in phase 2.2) for the trial trust radius $\bar{\Delta}_k$ satisfies $\rho_k^j(p_k^j) < \eta_2$, then we are *allowed* to execute step 2.3).

2.3) Find some $\Delta_k \in (0,\bar{\Delta}_k)$, model with index i, and $p_k = p_k^i \in A(i,\Delta_k)$ satisfying $\rho_k^i(p_k^i) \geq \eta_1$.

We now consider this algorithm in more detail.

4.5.1. Step 2.1 : Internal Doubling

No requirements are made on model switching at this stage. The initial model can be retained or any new model can be chosen, as long as the final choice of model, step, and trust radius satisfy both $p_k = p_k^i \in A(i,\Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$.

Step 2.1) is usually referred to as ''internal doubling'' because a typical implementation successively doubles $\bar{\Delta}_k$ as long as $\rho_k^i(p_k^i) \geq \eta_3$ for some $\eta_3 \in (\eta_2, 1)$. Another option this step allows is that of always trying a full quasi-Newton step before computing a constrained step.

Many implementations do not include steps of this type, and most of the algorithms we present here specify that such ''internal doubling'' steps are optional. However, this step makes many line search procedures admissible to our theory. Furthermore, inclusion of such logic is particularly appropriate for many model switching applications. Consider a secant algorithm[2] that recomputes the exact Hessian whenever the secant approximation leads to a suspiciously small trust radius. If refreshing the secant model with the exact Hessian does indeed improve the model significantly, we would expect that a *much* larger trust radius can be successfully used. Internal doubling allows the trust radius to increase immediately by a sequence of minor iterations[3] thus avoiding the extra gradient evaluations associated with a sequence of major iterations.

4.5.2. Step 2.2 : Acceptance of trial trust radius

If step 2.1) is not executed, step 2.2) must be tried before 2.3). Again there are no restrictions on how many or how few of the models we use to compute trial steps. A step is first computed using the initial trial model and trust radius. If this step satisfies $\rho_k^i(p_k^i) \geq \eta_1$, then it can be accepted and the algorithm can proceed to step 3). Other options are allowable, however. The algorithm can try switching models to see whether others perform better. It can revert to step 2.1) if a sufficiently good model can be found.

[2] Such as the MINPACK algorithm HYBRID of Moré, Garbow, and Hillstrom [1980].

[3] Each minor iteration involves only linear algebra and a function evaluation.

4.5.3. Step 2.3 : Internal decrease of trust radius

The *only* real restriction on the logic of AS.4.4 deals with the cases for which step 2.3) can and cannot be executed. It *can* be tried if in the course of step 2.2) *any* model is found for which $\rho_k^i(p_k^i) < \eta_2$. It *cannot* be tried if, so far, *every* p_k^i tried in step 2.2) satisfies $\rho_k^i(p_k^i) \geq \eta_2$. The only circumstance where it *must* be tried is when every available model has been used to compute a step p_k^i in step 2.2) and for each one $\rho_k^i(p_k^i) < \eta_1$.

These conditions are the most general ones allowed by our theory and are much more relaxed than most implementations allow. In practice an algorithm should try to avoid executing 2.3) whenever possible, since smaller trust regions slow convergence. On the other hand, excessive switching of models in step 2.2) leads to extra overhead and function evaluations. Any given implementation is a tradeoff between these two considerations. Typically, the trust radius is not reduced if the initial trial model and step satisfies $\rho_k^i(p_k^i) \geq \eta_1$, and no alternative model is tried if $\rho_k^i(p_k^i) \geq \eta_2$.

If step 2.3) is executed, it must contain a procedure for actually calculating a reduced trust radius and step. The following is a version of the backtracking procedure typically used in single model trust region algorithms. This procedure is what gives Algorithm AS.4.4 the name ''internal loop.''

Procedure A.4.5 : Internal Decrease of Trust Radius

Let $0<\gamma_1<1$, $\eta_1 \in (0,1)$, a trial trust radius $\bar{\Delta}_k$, and a set of N_k models all be given.

Initialize $\Delta_k \in (0, \bar{\Delta}_k]$.

2.3.1) Set $\tau \in (0, \gamma_1]$.

2.3.2) Set $\Delta_k = \tau \Delta_k$.

2.3.3) Choose a model with index i and compute $p_k^i \in A(i, \Delta_k)$.

If $\rho_k^i(p_k^i) \geq \eta_1$ then accept p_k^i as p_k and proceed to step 3).

If not, either repeat this step for another model (one that has not yet been used for this Δ_k), or return to step 2.3.1).

Some very important points should be made about this procedure. First, AS.4.3 allows this procedure to start with at $\Delta_k = \| p_k^i \|$ if $\| p_k^i \| \in (0, \bar{\Delta}_k]$. Such an initialization is often done in practice when the step computed in phase 2.2) is a full quasi-Newton step in the interior of the trust region.

Second, let us consider the possibility of switching model preference during Procedure A.4.5. This is an important practical point. It was found in the development of NL2SOL (DGW [1981]) that performing model switching during internal reduction significantly degraded performance, in that it added excessive overhead. It is thus important that our theory cover algorithms that do not perform switching at this time. On the other hand, a reasonable implementation might be to try such switching if a large number of internal reductions are done in a row (as is done in Dennis,Sheng,Vu [1985]). Our theory must cover this case also. Therefore Algorithm AS.4.4 neither requires nor prohibits switching of models within the loop to decrease Δ_k.

Third, for a set of quadratic models in standard form, this procedure terminates in a finite number of iterations. Consider the following conditions defining an "acceptable" step with respect to a model with Hessian B_k^i.

(A) $p_k^i \in A(i, \Delta_k)$ implies that, for some $c_3 \in (0,1)$, p_k^i satisfies

$$pred_k^i\ (p_k^i) \geq c_3 \; \| g_k \| \; \min\left\{ \Delta_k \, , \frac{\| g_k \|}{\| B_k^i \|} \right\} .$$

(B) $p_k^i \in A(i, \Delta_k)$ implies $\| p_k^i \| \leq (1 + \sigma_1) \Delta_k$ for some $\sigma_1 \in [0, 1)$ independent of k.

We then have the following.

Theorem 4.5 : *For a given $x_1 \in \mathbb{R}^n$, let $f : \mathbb{R}^n \to \mathbb{R}^1$ satisfy the standard assumptions. For a given $k \geq 1$, let x_k satisfy $x_k \in L(f, x_1)$, and let $\{ B_k^i \}, i=1,2, \ldots N_k$ be the set of Hessians of N_k quadratic models in standard form.*

If conditions (A) and (B) above hold, then Procedure A.4.5 terminates in a finite number of steps.

Proof : From Lemma 4.2, there exists a $\delta > 0$ such that $\Delta_k \leq \delta$ and $p_k^i \in A(i, \Delta_k)$ imply $\rho_k^i(p_k^i) \geq \eta_1$ for $i = 1,2, \ldots, N_k$. Now, if $\delta \geq \overline{\Delta}_k$ at the start of the procedure, $\rho_k^i(p_k^i) \geq \eta_1$ for every i. If not, then at most $\log(\frac{\delta}{\overline{\Delta}_k})/\log(\gamma_1)$ applications of 2.3.1) are needed to imply $\rho_k^i(p_k^i) \geq \eta_1$ for every i, and at most N_k executions of 2.3.3) can be performed per application of 2.3.1). Since $\gamma_1 < 1$, this implies our result. □

This result cannot be taken for granted in algorithms that allow for more general models. Any alternative to Procedure A.4.5 is acceptable as long as it obtains a final result that satisfies $0 < \Delta_k < \bar{\Delta}_k$, $p_k = p_k^i \in A(i, \Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$ for some model with index i.

4.6. "External" Modification of the Trust Radius

We now define the allowable logic for step 3) of A.4.3.

Algorithm Specification AS.4.6 : Permissible Choices in Step 3) at the k<u>th</u> Iteration

Let $\gamma_3 \geq 1$, $0 < \eta_2 < 1$ be given.

```
3) Choose a new trial trust radius for the next iteration :
    3.1) Do one of the following :
        3.1.1) Choose an increase factor $\tau \in [1, \gamma_3]$,
        3.1.2) Or, if phase 2.1) was *not* executed , and if for *some* model and
            associated step $p_k^j \in A(j, \bar{\Delta}_k)$ from phase 2.2) or $p_k^j \in A(j, \Delta_k)$ from
            phase 2.3) we have $\rho_k^j(p_k^j) < \eta_2$, then we *are allowed* to choose a
            decrease factor $\tau \in (0,1]$.
    3.2) Do one of the following :
        3.2.1 ) Set the new trial radius, $\bar{\Delta}_{k+1}$, to $\tau\Delta_k$
        3.2.2 ) If, and only if, $p_k^i$ satisfies the inexact quasi-Newton condition
            for its associated model with Hessian $B_k^i$, then we *are allowed to*
            set $\bar{\Delta}_{k+1}$ to $\tau \| p_k^i \|$.
Proceed.
```

Step 3) also allows a very broad range of actions. We are always allowed to set the new trial trust radius to be larger (up to a constant factor) than the current radius. There are two instances when it can be chosen smaller than the current value. If $\rho_k^j(p_k^j) < \eta_2$ for some step p_k^j computed in 2.2), then the next trial radius can be set to any value greater than 0. Also, if the step finally chosen satisfies the inexact quasi-Newton condition (4.6b), then we can set the new trial trust radius so that $\bar{\Delta}_{k+1} \in [\, \| p_k \| \,, \gamma_3 \| p_k \| \,]$. This could be less than the current radius Δ_k since $\| p_k \|$ may be less than Δ_k.

Finally, the new trial radius can be set to *any* value greater than zero if any step we calculated using some model with index j at any stage of step 2.2) or 2.3) gave $\rho_k^j(p_k^j) < \eta_2$ unless the actual step p_k^j accepted came from 2.1).

4.7. Global Convergence Theory

We now state the main theorem of this chapter. The proof of this theorem is similar to a SSB [1985] result for single model algorithms. However, provisions are made to allow inexact quasi-Newton steps as well as exact ones to be used for trust radius modification. Also, our proof does not impose any conditions on *how* the algorithm executes an internal or external trust radius reduction, so long as this reduction is allowed. This, combined with the fact that it applies to multiple models, makes it a significant generalization of existing theory.

Consider the following conditions.

(A) Every model is a quadratic in standard form.

(B) $p_k^i \in A(i, \Delta_k)$ implies that, for some $c_3 \in (0,1)$ independent of k, p_k^i satisfies

$$pred_k^i\ (p_k^i) \geq c_3 \ \| g_k \| \ \min \left\{ \Delta_k \ , \ \frac{\| g_k \|}{\| B_k^i \|} \right\} .$$

(C) $p_k^i \in A(i, \Delta_k)$ implies $\| p_k^i \| \leq (1 + \sigma_1) \Delta_k$ for some $\sigma_1 \in [0, 1)$ independent of k.

(D) There exists $\beta_2 > 0$ independent of k such that $\| B_k^i \| \leq \beta_2$ for $k = 1, 2, ...$ and $i = 1, 2, ..., N_k$.

We then have the following.

Theorem 4.6 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Suppose that an algorithm implemented consistently with A.4.3, AS.4.4, and AS.4.6 is applied to f starting from x_1, and that the conditions (A), (B), and (C) hold. Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Then the algorithm generates a sequence $\{x_k\}$ which has the descent property.*

Furthermore, if (D) holds, then $\{g_k\}$ converges to zero.

Proof : This proof is in three parts. In (1) we establish the existence of $\{x_k\}$ and the descent of $\{f(x_k)\}$. In (2) we assume that $\{x_k\}$ remains in Ω_m (defined in (4.19)) for all $k \geq m$ and show that this leads to a contradiction. In (3) we prove the main result by appealing to Theorem 4.4.

(1) We first show that a sequence $\{p_k\}$ satisfying the conditions of the algorithm exists and has the descent property. Since $N_k < \infty$ the algorithm will eventually either find a satisfactory step p_k with $\Delta_k \geq \bar{\Delta}_k$, or proceed to step 2.3) for trust radius reduction. In the latter case, consider any $k \geq 1$ for which $x_k \in L(f, x_1)$. Since every model is a standard quadratic, Lemma 4.2 establishes that there exists a $\delta > 0$ for which $\Delta_k \leq \delta$ and $p_k^i \in A(i, \Delta_k)$ imply $\rho_k^i(p_k^i) > \eta_1$. Thus for Δ_k sufficiently small, the algorithm generates a satisfactory step p_k.

Furthermore, any step $p_k = p_k^i$ accepted by the algorithm must satisfy the condition $\rho_k^i(p_k^i) \geq \eta_1$, which with (B) implies $ared_k(p_k) > 0$. But $x_1 \in L(f, x_1)$, so that, by induction, $\{x_k\}$ exists and satisfies the descent condition.

(2) Let Ω_m be defined as in (4.19) and consider any $m \geq 1$ with $\| g_m \| \neq 0$. We want to show that some later iterate is not in Ω_m. First, we will establish some properties for any $x_k \in \Omega_m$.

Lemma 4.3 implies

$$\| g_k \| \geq \frac{1}{2} \| g_m \| , \tag{4.30}$$

so that from (B) and (D), we have

$$\begin{aligned} pred_k^i(p_k^i) &\geq c_3 \, \| g_k \| \, \min \left\{ \Delta_k , \frac{\| g_k \|}{\beta_2} \right\} \\ &\geq \frac{1}{2} c_3 \, \| g_m \| \, \min \left\{ \Delta_k , \frac{\| g_m \|}{2 \beta_2} \right\} . \end{aligned} \tag{4.31}$$

From (4.31), (A), and Lemma 2.6 we have

$$| 1-\rho_k^i(p_k^i) | = \left| \frac{ared_k(p_k^i) - pred_k^i(p_k^i)}{pred_k^i(p_k^i)} \right|$$

$$\leq \left| \frac{\int_0^1 (p_k^i)^t (B_k^i - H(x_k + \xi p_k^i))(p_k^i) (1 - \xi) \, d\xi}{\frac{1}{2} c_3 \| g_m \| \min \left\{ \Delta_k \, , \, \frac{\| g_m \|}{2\beta_2} \right\}} \right| . \tag{4.32}$$

Including (C), (D), and (SA3) in (4.32) yields

$$| 1-\rho_k^i(p_k^i) | \leq \frac{(1+\sigma_1)^2 \Delta_k^2 (\beta_1 + \beta_2)}{c_3 \| g_m \| \min \left\{ \Delta_k \, , \, \frac{\| g_m \|}{2\beta_2} \right\}} . \tag{4.33}$$

Let the scalars δ_m and δ_m^+ be defined as

$$\delta_m \equiv \frac{\| g_m \|}{4\gamma_3 (1+\sigma_1)^3} \; \frac{c_3 (1-\eta_2)}{\beta_1+\beta_2} \tag{4.34}$$

and

$$\begin{aligned} \delta_m^+ &\equiv (1+\sigma_1) \gamma_3 \, \delta_m \\ &= \frac{c_3 \| g_m \|}{4 (1+\sigma_1)^2} \; \frac{1-\eta_2}{\beta_1+\beta_2} . \end{aligned} \tag{4.35}$$

Consider any $\Delta_k < \delta_m^+$. Since $c_3 < 1$, from (4.35) we have $\Delta_k < \frac{\| g_m \|}{2\beta_2}$, so (4.33) implies

$$\begin{aligned} | 1-\rho_k^i(p_k^i) | &< \frac{(1 + \sigma_1)^2 \Delta_k (\beta_1 + \beta_2)}{c_3 \| g_m \|} . \\ &< \frac{(1+\sigma_1)^2}{c_3 \| g_m \|} (\beta_1 + \beta_2) \, \delta_m^+ . \end{aligned} \tag{4.36}$$

Hence from (4.34), for any $\Delta_k < \delta_m^+$ and $x_k \in \Omega_m$,

$$| 1-\rho_k^i(p_k^i) | < \frac{1}{4}(1-\eta_2) < 1-\eta_2 \tag{4.37}$$

and

$$2-\eta_2 > \rho_k^i(p_k^i) > \eta_2 \ . \tag{4.38}$$

Now, suppose that for some m, $\| g_m \| \neq 0$ and $x_k \in \Omega_m$ for all $k \geq m$. We establish that this assumption leads to a contradiction.

(2a) By supposition, $\| g_m \| \neq 0$ and $x_k \in \Omega_m$ for all $k \geq m$. Suppose further that $\lim_{k \to \infty} \sup \Delta_k < \delta_m$. Let $\hat{k}$ be the smallest integer for which $\Delta_k < \delta_m$ for all $k \geq \hat{k}$. Let ζ_k be defined as in (4.6b). Assume without loss of generality that $\hat{k} > m$ and that if option 3.2.2) is being used in AS.4.6, then $\hat{k}$ is sufficiently large to ensure $\zeta_k < \frac{1}{2}$ for all $k \geq \hat{k}$.

For any $\Delta_k < \delta_m$, AS.4.6 and (C) imply

$$\begin{aligned} \bar{\Delta}_k &\leq \gamma_3 \max \{ \Delta_{k-1}, \| p_{k-1} \| \} \leq \gamma_3 (1 + \sigma_1) \Delta_{k-1} \\ &< \gamma_3 (1 + \sigma_1) \delta_m \leq \delta_m^+ \ . \end{aligned} \tag{4.39}$$

Hence for every $k > \hat{k}$, (4.38) implies that any $p_k^i \in A(i, \bar{\Delta}_k)$ computed in phase 2.2) satisfies $\rho_k^i(p_k^i) > \eta_2$, so that phase 2.3) cannot be executed. Therefore $\Delta_k \geq \bar{\Delta}_k$ for every $k > \hat{k}$.

Now consider the computation of $\bar{\Delta}_{k+1}$ as specified by AS.4.6. We will show that if $x_k \in \Omega_m$, $k > \hat{k}$, then $\bar{\Delta}_{k+1} \geq \Delta_k$. Since $\Delta_k \leq \delta_m$, (4.38) and the definition of the algorithm imply $\tau \geq 1$. If the trial trust radius for the next iteration is computed by phase 3.2.1) of AS.4.6, then

$$\bar{\Delta}_{k+1} = \tau \Delta_k \geq \Delta_k \quad . \tag{4.40}$$

If p_k^i is a quasi-Newton step or an inexact quasi-Newton step for some model, we have from (4.6b)

$$\begin{aligned} \| B_k^i \| \, \| p_k^i \| &\geq \| B_k^i p_k^i \| = \| g_k - (B_k^i p_k^i + g_k) \| \\ &\geq \| g_k \| - \| B_k^i p_k^i + g_k \| \\ &\geq (1 - \zeta_k) \| g_k \| \end{aligned} \tag{4.41}$$

and

$$\begin{aligned} \| p_k^i \| &\geq \frac{(1 - \zeta_k) \| g_k \|}{\| B_k^i \|} \geq \frac{\| g_k \|}{2 \beta_2} \geq \frac{\| g_m \|}{4 \beta_2} \\ &> \delta_m > \Delta_k \quad . \end{aligned} \tag{4.42}$$

If the trial trust radius for the next iteration is computed by phase 3.2.2) of AS.4.6, (4.42) implies

$$\bar{\Delta}_{k+1} = \tau \| p_k^i \| \geq \Delta_k \tag{4.43}$$

so that in either event $\bar{\Delta}_{k+1} \geq \Delta_k$ for every $k > \hat{k}$.

Let $\delta_m^- \equiv \bar{\Delta}_{\hat{k}+1}$. Since $\Delta_k \geq \bar{\Delta}_k$ and $\bar{\Delta}_k \geq \Delta_{k-1}$ for all $k > \hat{k}$,

$$\therefore \qquad \liminf_{k \to \infty} \Delta_k \;\geq\; \delta_m^- > 0 \quad . \tag{4.44}$$

Hence, if our supposition is true, either

$$\delta_m \;\ge\; \limsup_{k\to\infty}\Delta_k \;\ge\; \liminf_{k\to\infty}\Delta_k \;\ge\; \delta_{\bar{m}} \;>\; 0 \tag{4.45}$$

or

$$\limsup_{k\to\infty}\Delta_k \;\ge\; \delta_m \;>\; 0. \tag{4.46}$$

In either event, $\limsup_{k\to\infty}\Delta_k \;>\; 0.$

(2b) By supposition, $\| g_m \| \neq 0$ and $x_k \in \Omega_m$ for all $k \geq m$. Conditions (B), (D) and (4.31) imply

$$\begin{aligned} f_k - f_{k+1} &= ared_k(p_k^i) \ge \eta_1\, pred_k^i(p_k^i) \\ &\ge \frac{1}{2}\eta_1\, c_3 \,\| g_m \| \,\min\left\{ \Delta_k \;,\; \frac{\| g_m \|}{2\beta_2} \right\} . \end{aligned} \tag{4.47}$$

By the standard assumptions, f is bounded below so that the descent condition implies $\lim_{k\to\infty}(f_k - f_{k+1}) = 0$. Therefore $\lim_{k\to\infty}\Delta_k = 0$, which contradicts (2a).

Hence, for every m with $\| g_m \| \neq 0$, there exists $\bar{m} > m$ for which $x_{\bar{m}} \notin \Omega_m$.

(3) Part (1) showed that $\{x_k\}$ remains inside $L(f, x_1)$, and Part (2) implied that for every m with $\| g_m \| \neq 0$, there exists $\bar{m} > m$ for which $x_{\bar{m}} \notin \Omega_m$. Hence, eventually some $x_{\bar{m}}$ must satisfy

$$\| x_{\bar{m}} - x_m \| \ge \frac{\| g_m \|}{2\beta_1} \quad . \tag{4.48}$$

Therefore, Theorem 4.4 implies that $\{g_k\}$ converges to zero. □

Corollary 4.7 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. If conditions*

(A), (B), and (C) hold, then the sequences $\{x_k\}$ *generated by A.4.1 and A.4.2 starting from* x_1 *exist and are FOSPC.*

Proof : The standard assumptions on f and the definitions of A.4.1 and A.4.2 imply that each model satisfies conditions (A) and (D). Existence and FOSPC of $\{x_k\}$ then follow directly from application of Theorem 4.5 and Theorem 4.6 to A.4.1 and A.4.2. □

4.8. Other Allowable Options

We also present a variation of AS.4.6 that has one more allowable option at the cost of one more restriction. Both of these different details are commonly found in trust region implementations.

Algorithm Specification AS.4.7 : Permissible Choices in Step 3) at the k$^{\underline{\text{th}}}$ Iteration

Let $0 < \eta_1 < \eta_2 < \eta_3 < 1$, $\sigma_1 \in [\,0,1)$, and $1 < \dfrac{1}{1-\sigma_1} \leq \gamma_2 \leq \gamma_3$.

```
3) Choose a new trial trust radius for the next iteration :
    3.1) If $\rho_k^i(p_k^i) > \eta_3$ for the $p_k^i$ selected in step 2.2) then we are *required* to
            choose $\tau \in [\gamma_2,\gamma_3]$,
        Else
        3.1.1) Either choose an increase factor $\tau \in [1,\gamma_3]$,
        3.1.2) Or, if phase 2.1) was *not* executed , and if for *some* model and
            associated step $p_k^j \in A(j,\bar{\Delta}_k)$ from phase 2.2) or $p_k^j \in A(j,\Delta_k)$ from
            phase 2.3) we have $\rho_k^j(p_k^j) < \eta_2$, then we *are allowed* to choose a
            decrease factor $\tau \in (0,1]$.
    3.2) Set the new trial radius to either $\tau\Delta_k$ or $\tau\|p_k^i\|$.

Proceed.
```

The logic in 3.1) forces the use of an increase factor τ under certain conditions. Because of this, the new trial trust radius can be based on either the length of the last step or the size of the last trust radius. In AS.4.6, care must be taken if the norm of the current step is used to calculate the next trial radius because if p_k lies on the interior of the trust region, the trust region may be reduced even if no decrease factor was applied.

Consider the following conditions. Only the condition (C) differs from the assumptions of the previous theory.

(A) Every model is a quadratic in standard form.

(B) $p_k^i \in A(i, \Delta_k)$ implies that, for some $c_3 \in (0,1)$, p_k^i satisfies

$$pred_k^i\,(p_k^i) \geq c_3 \,||\, g_k \,||\, \min\left\{ \Delta_k \,,\, \frac{||\, g_k \,||}{||\, B_k^i \,||} \right\} .$$

(C) $p_k^i \in A(i, \Delta_k)$ implies condition (4.7 a or b) or (4.8 a or b) for some $\sigma_1 \in [0, 1)$ independent of k.

(D) There exists $\beta_2 > 0$ independent of k such that $||\, B_k^i \,|| \leq \beta_2$ for $k=1,2,$ ... , $i=1,2, \cdots ,N_k$.

Theorem 4.8 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Suppose that an algorithm implemented consistently with A.4.3, AS.4.4, and AS.4.7 is applied to f starting from x_1, and that the conditions (A), (B), and (C) hold. Then the algorithm generates a sequence $\{x_k\}$ which has the descent property. Furthermore, if (D) holds, then $\{g_k\}$ converges to zero.*

Proof : The proof is almost the same as that for Theorem 4.6. AS.4.7 differs from AS.4.6 only in the trust radius modification logic, and hence only section (2) of the proof need be modified. We need only make the following changes.

(1) Substitute η_3 for η_2 throughout the proof.

(2) In the third paragraph of (2a), notice that since $\Delta_k \leq \delta_m$, (4.38) implies that $\rho_k^i(p_k^i) \geq \eta_3$. Therefore AS.4.7 requires the selection of a τ satisfying

$$\tau \geq \gamma_2 \geq (1-\sigma_1)^{-1} \quad . \tag{4.49}$$

Now, AS.4.7 always allows $\bar{\Delta}_{k+1}$, to be defined by $\tau \| p_k^i \|$ as in (4.43), but (C) implies that p_k^i either satisfies the inexact quasi-Newton condition, or

$$\| p_k^i \| \geq (1 - \sigma_1) \Delta_k \quad . \tag{4.50}$$

In either event, (4.42), (4.49), and (4.50) imply

$$\begin{aligned} \bar{\Delta}_{k+1} &= \tau \| p_k^i \| \geq \gamma_2 (1 - \sigma_1) \Delta_k \\ &\geq \Delta_k \end{aligned} \tag{4.51}$$

for every $k > \hat{k}$; hence (4.43) through (4.46) are true for AS.4.7 as well as AS.4.6. This establishes the result. □

4.9. Summary : Type One Algorithms

Definition 4.3 : We refer to any algorithm of the form A.4.3 implemented consistently AS.4.4 and AS.4.6 or AS.4.7 as a **type one** algorithm.

This class of algorithms puts no limitations on how model switching or selection can be done, and puts few limitations on the permissible range of trust region logic. However, each of the models must be quadratic with

$$\limsup_{k \to \infty} \| B_k^i \| < \infty \quad .$$

It should be noted that the number of models available at any given iteration is not required to be constant. In fact, any finite number of models could be used. For example, consider the model generation scheme of Nazareth [1980,1983] that selects a particular convex combination of two model Hessians. A modification could be made to Nazareth's algorithm so that it would consider more than one possible convex combination. Our theory allows the algorithm to draw any number of model

Hessians from the infinite set of possible models as long as the sampling procedure is finite. If there exists a uniform upper bound on the curvature of each of these models, then the algorithms of this chapter are FOSPC.

CHAPTER 5

Safe Algorithms using fewer restrictions on the various models

5.1. Introduction

In this chapter, we assume that at least one model is a quadratic in standard form with uniformly bounded curvature. The other models and associated trial steps are only assumed to obey a ``uniform predicted decrease'' condition. We present model switching algorithms that are FOSPC under these assumptions. These algorithms are computationally efficient, even if implemented with a large number of models. Furthermore, no unnecessary assumptions are made about the alternative models. Specifically, these models need not be quadratic, and they are not even required to generate steps which are in descent directions. These algorithms and the theory describing them thus represent a significant advance in the understanding of this area.

5.2. Preliminary Definitions

We use much of the same nomenclature as in Chapter 4. Standard quadratic models are defined as in (4.1). Now, however, $pred_k^i(p)$ may refer to a general model which need not be quadratic. The actual function reduction is defined by (4.10), and $\rho_k^i(p)$ is defined as in (4.11).

Definition 5.1 : A set of models $\{pred_k^i\}$, $k=1,2,\ldots$, $i=1,2,\ldots$, N_k and associated set of steps $\{p_k^i\}$ is said to satisfy the **uniform predicted decrease condition** (**UPD**) if there exist $c_3 \in (0,1)$ and $\beta_3 \in [0,\infty)$ such that

$$pred_k^i\ (p_k^i) \geq c_3 \, \|g_k\| \, \min\left\{ \Delta_k \, , \, \frac{\|g_k\|}{\beta_3} \right\} \quad . \tag{5.1}$$

The UPD condition is a statement about a sequence of models and the associated procedures for computing trial steps. It is *not* a requirement on the function, or on how well the model matches the function.

In Chapter 4, all models were assumed to be standard quadratics with uniformly bounded curvature. These assumptions were used in two different ways in our proofs. First, they were sufficient to ensure that FCD and OLC steps satisfied the UPD condition given by equation (4.5). They were also sufficient to ensure that FOSPC is not prevented by spurious reduction of the trust radius. The algorithms of this chapter still require all trial steps to satisfy the UPD condition, now denoted by (5.1). However, by linking the trust radius reduction logic with the model switching logic, we show that the inclusion of a single standard quadratic model with uniformly bounded curvature is a sufficient condition to prevent spurious trust radius reduction.

For standard quadratic models and algorithms which use FCD or OLC procedures to compute steps, conditions weaker than uniformly bounded curvature are sufficient to imply the UPD condition. In Chapter 6, we will present some of these conditions.

An interesting point is that no requirement is made that p_k^i be a descent direction, as long as the UPD condition is satisfied. For example, some of the local minimizers of problem TRS in the indefinite case may be in a direction which initially increases the function, yet which still result in a sufficient amount of function reduction.

We will make use of the following definitions. First, recall the *e-test* from Chapter 3. The observed error for a given model and test step is defined as

$$e_k^i(p) \equiv | \ ared_k(p) - pred_k^i(p) \ | \quad . \tag{5.2}$$

The set of indices to models available at a given iteration k is defined as

$$I_k \equiv \left\{ i \ : \ 1 \leq i \leq N_k \right\} \quad . \tag{5.3}$$

Any arbitrary partition of I_k will be represented by two sets of indices denoted $\tilde{I}_k$ and $\hat{I}_k$ which satisfy

$$\tilde{I}_k \cup \hat{I}_k = I_k \tag{5.4}$$

and

$$\tilde{I}_k \cap \hat{I}_k = empty \quad . \tag{5.5}$$

5.3. Introductory Examples

Under the assumption that using a constant model Hessian is sufficiently less expensive to compensate for its slower convergence, Algorithm A.4.1 of the last chapter used a constant matrix as the primary model and the exact Hessian as a backup. In addition to being inexpensive to work with, constant Hessian approximations are natural candidates for safeguarding other models.

Let model one be some arbitrary model of interest which is assumed to generate steps satisfying the UPD condition. For a constant symmetric matrix B, let model two be

$$pred_k^2(p) = -g_k^t p - \frac{1}{2} p^t B p \quad . \tag{5.6}$$

The following algorithm always tries model one first, but may change to the constant model if model one exhibits poor performance. The matrix B can be something as simple as the identity matrix.

Algorithm A.5.1 : Safeguarding with a Constant Model

```
0) Let $x_1 \in \mathbb{R}^n$, $\Delta_1 > 0$, and two models be given.

1) Repeat until convergence :

    2) Compute $p_k^1 \in A(1, \Delta_k)$.

    3) If $\rho_k^1(p_k^1) < 0.01$, then
         Compute $p_k^2 \in A(2, \Delta_k)$.
         If $\rho_k^2(p_k^2) < 0.01$, then
              Set $\Delta_k = \Delta_k / 4$ and return to 2),
         Else set $p_k = p_k^2$, $\rho_k = \rho_k^2(p_k^2)$,
       Else
         Set $p_k = p_k^1$, $\rho_k = \rho_k^1(p_k^1)$.
       End if.

    4) Set $\Delta_{k+1} = \Delta_k$.
       If $\rho_k \geq 0.5$ set $\Delta_{k+1} = 2\Delta_k$.

    5) Set $x_{k+1} = x_k + p_k$ and define model 1 for the next iteration.

6) End
```

Rather than defining the constant model to be the primary model, this algorithm always uses it as the backup. The backup model is now assigned the function of making the algorithm safe rather than accelerating it. This is an example of the capability of multi-model algorithms to safeguard unproven methods. Notice that this algorithm will only reject model one if it is performing poorly already *and* model two looks better. A badly scaled choice of the "safe" model (such as the identity matrix) will not cause the performance degradation seen in Algorithm A.3.2. Notice also that no requirement is made concerning whether or not a failure of the primary model results in it being "reset" to the value of the backup.

Algorithm A.5.1 has to perform two function evaluations before any trust radius reduction can be done. Since such reductions are not uncommon, we now give a less expensive algorithm.

Algorithm A.5.2 : Safeguarding a method with a Constant Model

```
0) Let $x_1 \in \mathbb{R}^n$, $\Delta_1 > 0$, and two models be given.
1) Repeat until convergence :

    2) Compute $p_k^1 \in A(1, \Delta_k)$.

    3) If $\rho_k^1(p_k^1) < 0.01$ and $e_k^1(p_k^1) \geq 5 e_k^2(p_k^1)$, then
            Compute $p_k^2 \in A(2, \Delta_k)$.
            If $\rho_k^2(p_k^2) < 0.01$, then
                Set $\Delta_k = \Delta_k / 4$ and return to 2),
            Else Set $p_k = p_k^2$, $\rho_k = \rho_k^2(p_k^2)$.
       Else,
            Set $p_k = p_k^1$, $\rho_k = \rho_k^1(p_k^1)$.
       End if.

    4) Set $\Delta_{k+1} = \Delta_k$.

       If $\rho_k \geq 0.5$ set $\Delta_{k+1} = 2\Delta_k$.

    5) Set $x_{k+1} = x_k + p_k$ and define model 1 for the next iteration.

6) End
```

This algorithm is identical to A.5.1 except for the added requirement in step 3). However, it can often avoid extra function evaluations. Specifically, even if the primary model is performing poorly, we need not compute and test a step from the alternative model unless it does a *significantly* better job of prediction for the current trial step than the primary model. Thus this algorithm is highly biased against the use of the secondary model, even beyond the fact that the primary model is always tried first.

Because of the its low additional overhead, its high bias toward using the primary model, and its insensitivity to the possibility of a poorly scaled secondary model, this algorithm is exceedingly well suited to safeguarding new or unproven methods. Again, no requirements are made on the primary model other than the UPD condition. A less simplified version of this algorithm is suggested for general use.

A version of A.5.2 could also be used as a method of deciding when to "refresh" a secant model with an exact Hessian by using $pred_k^2(p) = -g_k^t p - \frac{1}{2} p^t H_k p$. To avoid evaluating H unnecessarily when performing the $e-test$, $pred_k^2(p_k^1)$ should be approximated by finite differences.

5.4. The Basic Algorithm Framework

We now present algorithms for model switching when at least one model is a standard quadratic with uniformly bounded curvature and all of the models satisfy the UPD condition. Algorithm A.4.3 is still used for the basic framework.

5.5. Computation of an Acceptable Step

The following specifications are sufficient to define the range of allowable options in step 2) of A.4.3. They differ from those given for type one algorithms only in their criteria for allowing trust radius reduction.

Algorithm Specification AS.5.3 : Permissible Choices in Step 2 of A.4.3 at the k$^{\underline{th}}$ Iteration

Let $\mu > 0$, $0 < \eta_1 \leq \eta_2 < 1$, a trial trust radius $\bar{\Delta}_k > 0$, and a set of N_k models be given.

2.1) We are *allowed* to accept $\Delta_k > \bar{\Delta}_k$ and $p_k = p_k^i$ for some model with index i *whenever* such a radius, model, and step satisfying both $p_k = p_k^i \in A(i, \Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$ can be found.

2.2) We are *allowed* to accept $\Delta_k = \bar{\Delta}_k$ and $p_k = p_k^i$ for some model with index i *whenever* both $p_k^i \in A(i, \Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$.

We are *allowed* to go to step 2.3) for trust radius reduction *only if* one of the following holds for the trial steps computed in 2.2) :

Reduction_Test_0 : *Every* model with index $i \in I_k$ and associated trial step $p_k^i \in \bar{A}(i, \bar{\Delta}_k)$ for the trial trust radius $\bar{\Delta}_k$ satisfied $\rho_k^i(p_k^i) < \eta_2$, or if

Reduction_Test_1 : $\rho_k^i(p_k^i) < \eta_2$ for $i \in \hat{I}_k$, and for every $j \in \tilde{I}_k$ $\exists$ at least one $i \in \hat{I}_k$ such that $\mu e_k^j(p_k^i) > e_k^i(p_k^i)$.

2.3) Find some $\Delta_k \in (0, \bar{\Delta}_k)$, model with index i, and $p_k = p_k^i \in A(i, \Delta_k)$ satisfying $\rho_k^i(p_k^i) \geq \eta_1$.

In Reduction_Test_0, I_k is the set of all models. In an implementation of Reduction_Test_1, $\hat{I}_k$ refers to models for which we have already computed p_k^i and $ared_k(p_k^i)$, while $\tilde{I}_k$ is the set of models for which we have not done this computation. These two tests are often easier to apply if stated to specify when trust radius reduction is *not* permissible. In Reduction_Test_0, reduction is *not* allowed if there exists a step $i \in I_k$ which satisfies $\rho_k^i(p_k^i) \geq \eta_2$. In Reduction_Test_1, reduction is *not* allowed if *either* of the following hold.

(i) There exists a step $i \in \hat{I}_k$ which satisfies $\rho_k^i(p_k^i) \geq \eta_2$, or

(ii) If $\rho_k^i(p_k^i) < \eta_2$ for all $i \in \hat{I}_k$, and there exists $j \in \tilde{I}_k$ with

$$e_k^j(p_k^i) \leq \frac{1}{\mu} e_k^i(p_k^i) \tag{5.7}$$

for all $i \in \hat{I}_k$.

We now consider this algorithm in more detail.

5.5.1. Step 2.1: Internal Doubling

This step is specified identically with type one algorithms.

5.5.2. Step 2.2 : Acceptance of Trial Trust Radius

The conditions for acceptance of a potential step p_k^i are the same as for type one algorithms. Any step found which satisfies $p_k^i \in A(i, \bar{\Delta}_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$ can be accepted. Even if such a step has been found, the algorithm is still allowed to compute the test steps from other models. However, the conditions under which we can reduce the trust radius by going to step (2.3) are more restrictive. Two alternative conditions are shown in AS.5.3 : Reduction_Test_0 and Reduction_Test_1.

Reduction_Test_0 prohibits reducing the trust radius if any of the available models generates a step which satisfies $\rho_k^i(p_k^i) \geq \eta_2$. Thus, in a three model system, even if $\rho_k^1(p_k^1) < \eta_1$ and $\rho_k^2(p_k^2) < \eta_1$, p_k^3 must still be computed and tested before any reduction is allowed. Reduction is only required if steps have been computed for all of the models and $\rho_k^i(p_k^i) < \eta_1$ for each. If $\eta_1 = \eta_2$, notice that the reduction is required if and only if it is allowed.

Algorithm A.5.1 is an example of logic based on this criteria. Even for this case where only two models are used, we see that Reduction_Test_0 criteria can lead to excessive switching.

Reduction_Test_1 is much more efficient. Say that we have computed p_k^i and found $\rho_k^1(p_k^1) < \eta_1$. Then we are allowed to reduce the trust radius unless one of the other models does a significantly better job of predicting the actual function value. The parameter μ can be adjusted to control the relative frequency between model switching and trust radius reduction in the algorithm's search for an acceptable step.

Algorithm A.5.2 is an example of logic based on Reduction_Test_1. Model switching is suppressed by setting μ to the relatively large value of 5.

Further discussion of condition Reduction_Test_1 is given in Section 5.7. Reduction_Test_0 is included here only as an elementary example of Reduction_Test_1, and is not recommended for actual use.

5.5.3. Step 2.3 : Internal Decrease of the Trust Radius

If the logic in step 2.2) allows an internal decrease, then we can use *any* procedure to do so, as long as the final step chosen satisfies $p_k = p_k^i \in A(i, \Delta_k)$ for some model with index i and $\Delta_k \in (0, \bar{\Delta}_k)$, and $\rho_k^i(p_k^i) \geq \eta_1$. No requirement is made on how the radius is reduced or whether or not the same model is retained. Therefore both the NL2SOL (DGW [1981]) tactic of retaining the same model throughout this phase and the Dennis, Sheng, Vu [1985] tactic of automatically switching after 5 unsuccessful decreases are admissible.

For quadratics (and most other reasonable models), Procedure A.4.5 is assured of finding an acceptable step in a finite number of minor iterations. This is not automatically true for all models. A trivial example is

$$pred_k^i(p) \equiv \frac{|g_k^i p|}{\eta_1} \tag{5.8}$$

which satisfies the UPD condition for any p yet does not satisfy $\rho_k^i(p) \geq \eta_1$ as $\|p\| \to 0$. Models of this sort are unlikely, but if they are included in the algorithm, Procedure A.4.5 will still find an acceptable step in a finite number of minor iterations if it switches models every time that a fixed number of internal decrease steps have been unsuccessful.

We give the following procedure for a two model algorithm with c the index of the current model and·a the index of the alternate. A new parameter is introduced: the integer $j_c \in [1, \infty)$. The function $mod(j, j_c)$ is the standard "modulus" operator that has value 0 if and only if $j = j_c$. The portions of the procedure involving j_c are included to guarantee successful termination of the inner reduction loop for general models and are are not essential if both models are quadratics. A suggested value for j_c is 5.

Procedure A.5.4 : Internal Decrease of the Trust Radius

```
Let $\gamma_1 \in (0,1)$, $\eta_1 \in (0,1]$ and $j_c \in [1,\infty)$ be given.

    For $j = 1,2,...$ do :

        Set $\Delta_k \in (0, \gamma_1 \Delta_k]$.
        Compute $p_k^c \in A(c, \Delta_k)$.
        If $\rho_k^c(p_k^c) \geq \eta_1$, then set $x_{k+1} = x_k + p_k^c$ and exit.

        If $mod(j, j_c) = 0$, then
            Switch model preference by switching the indices $a$ and $c$.
        End if.

    End for.
```

Again, this procedure is not required if both models are quadratics in standard form.

5.6. External Modification of the Trust Radius

The following specifications are sufficient to define the range of allowable options in step 3).

Algorithm Specification AS.5.5 : Permissible Choices in Step 3) of A.4.3 at the k$^{\underline{th}}$ Iteration

Let $\gamma_3 \geq 1$ be given.

```
3) Choose a new trial trust radius for the next iteration as follows :

3.1) Do one of the following :
    3.1.1) Choose an increase factor $\tau \in [1, \gamma_3]$,
    3.1.2) Or, if test Reduction_Test_0 or Reduction_Test_1 was satisfied at the
        last point of phase 2.2),
        then we are *allowed* to choose a decrease factor $\tau \in (0,1]$.

3.2) Set the new trial radius to $\tau\Delta_k$ .
```

As in type one algorithms, a trust radius increase by up to a constant factor is always allowed. If a trust radius decrease is allowed, it can be done in any fashion desired. However, the test allowing reduction is more stringent, in that $\rho_k^i(p_k^i) \leq \eta_2$ for the step selected may not be a sufficient condition to allow decrease if , for example, $\rho_k^j(p_k^j) \geq \eta_2$ for an alternative model.

5.7. Algorithm Efficiency and the Parameter μ

As stated previously, a multi-model algorithm must not require significantly more work per iteration than a single model algorithm. The parameter μ in Reduction_Test_1 allows us to control the amount of model switching in a very natural manner. As $\mu \rightarrow \infty$, the algorithm becomes indistinguishable from a model selection algorithm. As $\mu \rightarrow 0$, the algorithm becomes indistinguishable from one which always tests all models before allowing a trust region reduction.

NL2SOL uses $\mu=1.5$, which was experimentally found to be sufficiently large to prevent excessive switching, but small enough to keep the trust radius commensurate with the ``best'' model.[1] No preference is given to either model a priori (other than the hereditary preference), but μ can also be used to bias toward a model with a lower overhead.

Suppressing switching by choosing μ to be large is natural precisely because of the conditions under which it *allows* switching. A value of 100 will make model switching extremely rare, yet switch if the performance of the alternative model is undeniably superior.

The use of more than three possible models is somewhat speculative, but could occur in, say, an expert system code for optimization. Even for a large number of models, this type of algorithm is efficient. Consider $N_k=16$ and $\mu=2$. If the initial trial model yields $p_k^1(p_k^1)<\eta_2$, then the alternatives are scanned for those which are at least twice as good at predicting the actual reduction: $e_k^j(p_k^1) \le \frac{1}{2} e_k^1(p_k^1)$. Say, pessimistically, that models 2, 3, ..., 8 all satisfy this condition. Then models 9, 10, ..., 16 can be dropped from consideration, and we choose the remaining model with the smallest value of $e_k^j(p_k^1)$ to compute the next trial iterate. Now we scan models 3, 4, ..., 8 to find any which are twice as good as model two at predicting the actual

[1] This value offers support to stochastic arguments based on the central limit theorem which suggest a value of $\mu \ge \sqrt{N_k}$ when N_k models are available to the system. However, there are no computational results in existence for $N_k>2$, so this suggestion should be treated as speculative.

reduction. Again being pessimistic, models 3, 4 and 5 might be accepted, and assuming $e_k^3(p_k^2) < e_k^4(p_k^2)$, $e_k^5(p_k^2)$, we compute the potential iterate p_k^3. Being pessimistic a final time, suppose $e_k^4(p_k^3) \leq \frac{1}{2} e_k^3(p_k^3)$.

Now, it is unlikely that this point would have been reached, but suppose it has been. Then model 4 has predicted $ared_k(p_k^1)$ twice as well as model one, has predicted $ared_k(p_k^2)$ twice as well as model two (which itself was the best predictor for $ared_k(p_k^1)$), and now predicts $ared_k(p_k^3)$ twice as well as model three (which itself predicted $ared_k(p_k^2)$ very well). Under such conditions an algorithm would be unreasonable if it did *not* investigate model 4 before reducing the trust radius.

Thus, even for relatively large N_k and small μ, the algorithm should spend only as much effort in switching as is justified.

These arguments are somewhat hypothetical, but if logic is included in the switching algorithm to increase μ to a large value after two models have unsuccessfully computed steps, then it should be clear that excessive model switching during an iteration is not a valid concern.

It should not be assumed from this discussion that μ should always be chosen to be a large value. Such an approach is conservative with respect to making the most use of the different models. For any given application an appropriate μ should be found to ensure a reasonable tradeoff between alternatives.

5.8. Global Convergence Theory

Consider the following conditions.

(A) At every iteration k and for every $i \in I_k$, $p_k^i \in A(i, \Delta_k)$ implies that $pred_k^i\ (p_k^i) \geq c_3 \, \| g_k \| \min\{ \Delta_k \, , \| g_k \| / \beta_3 \}$ for some $c_3 \in (0,1)$ and $\beta_3 > 0$ independent of k.

(B) At every iteration k and for every $i \in I_k$, $p_k^i \in A(i, \Delta_k)$ implies $\| p_k^i \| \leq (1 + \sigma_1) \Delta_k$ for some $\sigma_1 \in [0, 1)$ independent of k.

(C) At every iteration k, $\exists q \in I_k$ such that $pred_k^q$ is a quadratic in standard form and $\| B_k^q \| \leq \beta_2$ for some $\beta_2 > 0$ independent of k.

We then have the following.

Theorem 5.1 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Suppose that an algorithm implemented consistently with A.4.3, AS.5.3, and AS.5.5 is applied to f starting from x_1, and let conditions (A), (B), and (C) hold. Then the algorithm generates a sequence $\{x_k\}$ which has the descent property. Furthermore, $g_k \rightarrow 0$.*

Proof : This proof is in three parts. In (1) we establish the existence of $\{x_k\}$ and the descent of $\{f(x_k)\}$. In (2) we assume that $\{x_k\}$ remains in Ω_m for all $k \geq m$ and show that this leads to a contradiction. In (3) we prove the main result by appealing to Theorem 4.4.

(1) We first show that a sequence $\{x_k\}$ satisfying the conditions of the algorithm exists and has the descent property. Since $N_k < \infty$ the algorithm will eventually either find a satisfactory step p_k with $\Delta_k \geq \bar{\Delta}_k$, or proceed to step 2.3) for trust radius reduction. In the latter case, consider any $k \geq 1$ for which $x_k \in L(f, x_1)$. Since at least one model at every iteration is a standard quadratic, Lemma 4.2 establishes that there exists a $\delta > 0$ and index i for which $\Delta_k \leq \delta$ and $p_k^i \in A(i, \Delta_k)$ imply $\rho_k^i(p_k^i) > \eta_1$. Thus for Δ_k sufficiently small, the algorithm can find[2] a satisfactory step p_k.

Furthermore, any step $p_k = p_k^i$ accepted by the algorithm must satisfy the condition $\rho_k^i(p_k^i) \geq \eta_1$, which with (A) implies

$$ared_k(p_k) \geq \eta_1 c_3 \, \| g_k \| \, \min \left\{ \Delta_k \, , \, \frac{\| g_k \|}{\beta_3} \right\} > 0 \quad . \tag{5.9}$$

But $x_1 \in L(f, x_1)$ so that, by induction, $\{x_k\}$ exists and satisfies the descent condition.

(2) Let Ω_m be defined as in (4.19), i.e.

$$\Omega_m \equiv \{ x : \| x - x_m \| < \frac{\| g_m \|}{2 \beta_1} \text{ and } x \in L(f, x_1) \} \, .$$

Consider any $m \geq 1$ with $\| g_m \| \neq 0$. We want to show that some later iterate x_k is not in Ω_m.

First, we will establish some properties for any $x_k \in \Omega_m$. Lemma 4.3 implies

[2] Recall that we did not specify *how* the algorithm would find p_k and i, but merely stated in step 2.3) that they are to be found. In this theorem we show that permissible values exist, and hence by the statement of the algorithm, they will be found. In practice, one would use an procedure similar to A.5.4, just as one would use Procedure A.4.5 with the algorithms of Chapter 4.

$$\| g_k \| \geq \frac{1}{2} \| g_m \| \tag{5.10}$$

so that (A) implies

$$\begin{aligned} pred_k^i(p_k^i) &\geq c_3 \, \| g_k \| \, \min \left\{ \Delta_k \, , \, \frac{\| g_k \|}{\beta_3} \right\} \\ &\geq \frac{1}{2} c_3 \, \| g_m \| \, \min \left\{ \Delta_k \, , \, \frac{\| g_m \|}{2 \beta_3} \right\} \quad . \end{aligned} \tag{5.11}$$

Let q be the index of a standard quadratic models satisfying condition (C). From Lemma 2.6, (B), (C), and (SA3)

$$\begin{aligned} e_k^q(p_k^i) &= | \, ared_k(p_k^i) - pred_k^q(p_k^i) \, | \\ &\leq \left| \int_0^1 (p_k^i)^t (\, B_k^q - H(x_k + \xi p_k^i) \,)(p_k^i) \, (\, 1 - \xi \,) \, d\xi \right| \\ &\leq \frac{1}{2} (\, 1 + \sigma_1 \,)^2 \, \Delta_k^2 \, (\, \beta_1 + \beta_2 \,) \quad . \end{aligned} \tag{5.12}$$

Let the scalars δ_m and δ_m^+ be defined as

$$\delta_m \equiv \frac{c_3 \, \| \, g_m \, \|}{2 \gamma_3 (\, 1 + \sigma_1 \,)^3 \, (\, \beta_1 + \beta_2 + \beta_3 \,)} \; \frac{(\, 1 - \eta_2 \,)}{\max\{ \, \mu \, , 1 \, \}} \tag{5.13}$$

and

$$\begin{aligned} \delta_m^+ &\equiv (\, 1 + \sigma_1 \,) \gamma_3 \, \delta_m \\ &= \frac{c_3 \, \| \, g_m \, \|}{2 \, (\, 1 + \sigma_1 \,)^2 \, (\, \beta_1 + \beta_2 + \beta_3 \,)} \; \frac{(\, 1 - \eta_2 \,)}{\max\{ \, \mu \, , 1 \, \}} \end{aligned} \tag{5.14}$$

While still assuming that $x_k \in \Omega_m$, let us further establish some results for any $\Delta_k < \delta_m^+$. Since $c_3 \leq 1$, (5.14) implies $\Delta_k < \dfrac{\| g_m \|}{2 \beta_3}$, so (5.11) reduces to

$$pred_k^i\,(p_k^i\,) \geq \frac{1}{2}\, c_3 \,\| g_m \|\, \Delta_k \quad . \tag{5.15}$$

Hence

$$\begin{aligned} |\, 1 - \rho_k^i\,(p_k^i\,)\,| &= \frac{e_k^i\,(p_k^i\,)}{|\, pred_k^i\,(p_k^i\,)\,|} \\ &\leq \frac{2\, e_k^i\,(p_k^i\,)}{c_3 \,\| g_m \|\, \Delta_k} \quad . \end{aligned} \tag{5.16}$$

Therefore for any $\Delta_k < \delta_m^+$ and $x_k \in \Omega_m$,

$$\begin{aligned} |\, 1 - \rho_k^q(p_k^q)\,| &< \frac{(\,1 + \sigma_1\,)^2}{c_3 \,\| g_m \|}\,(\,\beta_1 + \beta_2\,)\,\Delta_k \\ &< \frac{(\,1 + \sigma_1\,)^2}{c_3 \,\| g_m \|}(\,\beta_1 + \beta_2\,)\,\delta_m^+ \\ &< \frac{1}{2}(\,1 - \eta_2\,) < 1 - \eta_2 \end{aligned} \tag{5.17}$$

and

$$2 - \eta_2 > \rho_k^q(p_k^q) > \eta_2 \quad . \tag{5.18}$$

Now that we have established these results for any $\Delta_k < \delta_m^+$ and $x_k \in \Omega_m$, we can proceed with our contradiction.

Consider any $\| g_m \| \neq 0$ and suppose $x_k \in \Omega_m$ for all $k \geq m$.

(2a) By supposition, $\| g_m \| \neq 0$ and $x_k \in \Omega_m$ for all $k \geq m$. Suppose further that $\lim_{k \to \infty}$ $\sup \Delta_k < \delta_m$. Let $\hat{k}$ be the smallest integer for which $\hat{k} > m$ and $\Delta_k < \delta_m$ for all $k \geq \hat{k}$.

For any $\Delta_k < \delta_m$, AS.5.3 and (C) imply

$$\bar{\Delta}_k \leq \gamma_3 \max \{ \Delta_{k-1}, \|p_{k-1}\| \} \leq \gamma_3 (1 + \sigma_1) \Delta_{k-1} \tag{5.19}$$
$$< \gamma_3 (1 + \sigma_1) \delta_m \leq \delta_m^+ .$$

Hence (5.18) implies that for every $k > \hat{k}$, any $p_k^q \in A(q, \bar{\Delta}_k)$ computed in phase 2.2) satisfies

$$\rho_k^q(p_k^q) > \eta_2 . \tag{5.20}$$

Thus, if $q \in \hat{I}_k$, no reduction of the trust radius is allowed. Furthermore, if $q \in \tilde{I}_k$ and $\rho_k^i(p_k^i) \leq \eta_2$ for some $i \in \hat{I}_k$,

$$\frac{e_k^i(p_k^i)}{|pred_k^i(p_k^i)|} = | 1 - \rho_k^i(p_k^i) | \geq 1 - \eta_2 \tag{5.21}$$

and hence by (5.14), (5.15) and (5.19)

$$e_k^i(p_k^i) \geq (1 - \eta_2) | pred_k^i(p_k^i) | \geq \frac{1}{2} (1 - \eta_2) c_3 \|g_m\| \Delta_k$$
$$= (\beta_1 + \beta_2 + \beta_3) (1 + \sigma_1)^2 \max\{ \mu, 1 \} \delta_m^+ \Delta_k \tag{5.22}$$
$$> (\beta_1 + \beta_2 + \beta_3) (1 + \sigma_1)^2 \max \{ \mu, 1 \} \Delta_k^2$$

so that (5.12) implies

$$e_k^i(p_k^i) > 2 \max \{ \mu, 1 \} e_k^q(p_k^i) \geq \mu e_k^q(p_k^i) . \tag{5.23}$$

Therefore phase 2.3) cannot be executed, and hence $\Delta_k \geq \bar{\Delta}_k$ for every $k > \hat{k}$.

We next show that if $x_k \in \Omega_m$ for all $k > m$, then $\bar{\Delta}_{k+1} \geq \Delta_k$. Consider the computation of $\bar{\Delta}_{k+1}$ as specified by AS.5.5. Since $\Delta_k \leq \delta_m$, (5.20) and (5.23) and the definition of the algorithm imply $\tau \geq 1$. Therefore

$$\bar{\Delta}_{k+1} = \tau \Delta_k \geq \Delta_k \tag{5.24}$$

for every $k > \hat{k}$. Let $\delta_m^- \equiv \bar{\Delta}_{\hat{k}+1}$. Since $\Delta_k \geq \bar{\Delta}_k$ and $\bar{\Delta}_k \geq \Delta_{k-1}$ for all $k > \hat{k}$,

$$\liminf_{k \to \infty} \Delta_k \ \geq \ \delta_m^- > 0 \ . \tag{5.25}$$

Hence, either

$$\delta_m \ > \ \limsup_{k \to \infty} \Delta_k \ \geq \ \liminf_{k \to \infty} \Delta_k \ \geq \ \delta_m^- \ > \ 0 \tag{5.26}$$

or

$$\limsup_{k \to \infty} \Delta_k \ \geq \ \delta_m \ > \ 0. \tag{5.27}$$

In either event, $\limsup_{k \to \infty} \Delta_k \ > \ 0$.

(2b) By supposition, $\| g_m \| \neq 0$ and $x_k \in \Omega_m$ for all $k \geq m$. Conditions (5.9) and (5.10) imply

$$f_k - f_{k+1} = ared_k(p_k^i) \tag{5.28a}$$

$$\geq \eta_1 \, pred_k^i(p_k^i) \tag{5.28b}$$

$$\geq \frac{1}{2} \eta_1 \, c_3 \, \| g_m \| \, \min \left\{ \Delta_k \ , \ \frac{\| g_m \|}{2 \beta_3} \right\} . \tag{5.28c}$$

By the standard assumptions, f is bounded below so that the descent condition implies $\lim_{k \to \infty} (f_k - f_{k+1}) = 0$. Therefore $\lim_{k \to \infty} \Delta_k = 0$, which contradicts (2a).

Hence, for every m with $\| g_m \| \neq 0$, there exists $\bar{m} > m$ for which $x_{\bar{m}} \notin \Omega_m$.

(3) Part (1) of this proof showed that $\{x_k\}$ remains inside $L(f, x_1)$, and part (2) showed that for every m with $\| g_m \| \neq 0$, there exists $\bar{m} > m$ for which $x_{\bar{m}} \notin \Omega_m$.

Hence, eventually some $x_{\bar{m}}$ must satisfy

$$\| x_{\bar{m}} - x_m \| \geq \frac{\| g_m \|}{2\beta_1} \quad . \tag{5.29}$$

Therefore, Theorem 4.4 implies that $\{g_k\}$ converges to zero. □

5.9. Other Allowable Options

It is instructive to consider several slight variations to the previous algorithms. Two which will prove useful are as follows.

Algorithm Specification AS.5.6 : Permissible Modification to AS.5.3 and AS.5.5

Let $\bar{\delta}_0 > 0$ be given.

(a) Regardless of any of the previously specified logic, we are *allowed* to reduce the trust radius via step 2.3) or 3) at iteration k *whenever* $\bar{\Delta}_k > \bar{\delta}_0$. If $\bar{\Delta}_k \leq \bar{\delta}_0$, the previously specified logic must still be applied.

(b) Regardless of any of the previously specified logic, we are *allowed* to reduce the trust radius via step 2.3) or 3) at iteration k *whenever* $\bar{\Delta}_k > \bar{\delta}_0 \| g_l \|$ for some $l \leq k$, $g_l \neq 0$. If $\bar{\Delta}_k \leq \bar{\delta}_0 \| g_l \|$, the previously specified logic must still be applied.

Theorem 5.2 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Suppose that an algorithm implemented consistently with A.4.3, AS.5.3, AS.5.5, and modification AS.5.6 is applied to f starting from x_1, and let conditions (A), (B), and (C) hold. Then the algorithm generates a sequence $\{x_k\}$ which has the descent property. Furthermore, $g_k \rightarrow 0$.*

Proof : The proof of Theorem 5.1 need only be modified slightly to obtain these results. AS.5.6 differs from AS.5.3 and AS.5.5 only in the trust radius modification logic, and hence only section (2) of the proof is affected.

To establish the result given modification (a), we need merely define δ_m by

$$\delta_m \equiv \min\left\{ \frac{\bar{\delta}_0}{(1+\sigma_1)}, \frac{c_3 \| g_m \|}{2\gamma_3(1+\sigma_1)^3(\beta_1+\beta_2+\beta_3)} \frac{(1-\eta_2)}{\max\{\mu, 1\}} \right\} \tag{5.30}$$

rather than by (5.13). With (5.13) replaced by (5.30), the hypotheses in section (2a) of the proof imply that $\bar{\Delta}_k \le \bar{\delta}_0$. Hence, throughout section (2a), the trust radius modification logic of AS.5.3 and AS.5.5 must be used.

To establish the result given modification (b), define

$$\delta_m^{\#} = \frac{1}{2} \min_{\substack{k \le m \\ g_k \ne 0}} \| g_k \| \quad . \tag{5.31}$$

By hypothesis, for any $k \ge m$, $x_k \in \Omega_m$, so for any $l < k$, either

(i) $l \le m$ and $\| g_l \| \ge \delta_m^{\#}$, or

(ii) $l > m$ and $\| g_l \| > \frac{1}{2} \| g_m \| \ge \delta_m^{\#}$, so that $\bar{\delta}_0 \| g_l \| > \bar{\delta}_0 \delta_m^{\#} > 0$. We then replace (5.13) by

$$\delta_m \equiv \min\left\{ \frac{\bar{\delta}_0 \delta_m^{\#}}{\gamma_3(1+\sigma_1)}, \frac{c_3 \| g_m \|}{2\gamma_3(1+\sigma_1)^3(\beta_1+\beta_2+\beta_3)} \frac{(1-\eta_2)}{\max\{\mu, 1\}} \right\} \quad . \tag{5.32}$$

With (5.13) replaced by (5.32), the hypotheses in section (2a) of the proof imply that $\bar{\Delta}_k \leq \bar{\delta}_0 \| g_l \|$. Hence, throughout section (2a), the trust radius modification logic of AS.5.3 and AS.5.5 must be used. □

AS.5.6 is useful in the case where evaluating $e_k^j(p_k^j)$ for the model with index j is expensive. For example, in deciding between continuing with a secant model or resetting the model Hessian to the exact Hessian, performing an *e−test* for a trial step $\hat{p}$ involves either calculating the exact Hessian immediately or approximating the second directional derivative of f in the direction $\hat{p}$. Option (i) in AS.5.6 allows us to specify that the *e−test* is not to be done if the trust radius is greater than a specified minimum. Furthermore, consider any iteration l at which a restart is performed. Examination of (SA3), Lemma 4.2 and the inequality (4.42) applied to $B_k = H_k$ shows that if an internal doubling procedure is executed after each reset, the step selected must have norm greater than $\delta \| g_l \|$ for some $\delta > 0$ independent of k. Hence, option (ii) in AS.5.6 shows that an acceptable method for deciding when a reset is needed is to save the norm of the step taken at the last reset and execute a reset when the current trust radius drops beneath a fixed fraction of this norm.

Algorithm Specification AS.5.7 : Permissible modification to AS.5.3 and AS.5.5

In addition to the logic previously specified for step 2.2) of iteration k, we are *allowed* to accept $\Delta_k = \bar{\Delta}_k$ and $p_k \leq (1+\sigma_1)\Delta_k$ if there exists a model with index i and step $p_k^i \in A(i, \bar{\Delta}_k)$ for which $\rho_k^i(p_k^i) \geq \eta_1$ and $ared_k(p_k) \geq ared_k(p_k^i)$.

Theorem 5.3 : *Let $x_1 \in \mathbb{R}^n$ and f satisfy the standard assumptions. Suppose that an algorithm implemented consistently with A.4.3, AS.5.3, AS.5.5, and modification AS.5.7 is applied to f starting from x_1, and let conditions (A), (B), and (C) hold. Then the algorithm generates a sequence $\{x_k\}$ which has the descent property. Furthermore, $g_k \rightarrow 0$.*

Proof : To establish the result for modification (b), we need only change (5.28a) to

$$f_k - f_{k+1} = ared_k(p_k) \geq ared_k(p_k^i) \ . \tag{5.33}$$

The remainder of the proof is unchanged. □

AS.5.7 is motivated by the switching logic in NL2SOL. Let us denote the currently preferred model by "c" and the alternative by "a." The switching logic of NL2SOL is admissible under AS.5.3, AS.5.5 and AS.5.7 except in one instance, when we have

$$\rho_k^c(p_k^c) < \eta_2, \text{ and } e_k^c(p_k^c) > \mu\, e_k^a(p_k^c) \tag{5.34}$$

and

$$\rho_k^a(p_k^a) \geq \eta_2, \text{ and } ared_k(p_k^c) > ared_k(p_k^a). \tag{5.35}$$

Now, since NL2SOL selects its new model based on the largest actual reduction in this case, the current model $pred_k^c$ is retained. Furthermore, NL2SOL bases its decision about trust radius reduction on $\rho_k^c(p_k^c)$ even though $\rho_k^a(p_k^a) \geq \eta_2$ (which would prohibit trust radius reduction in our framework).

If $\eta_2 > \rho_k^c(p_k^c) \geq \eta_1$, then the logic of NL2SOL will reduce the trial radius before the next iteration, but will also try the alternate model first at the next iteration since $e_k^c(p_k^c) > \mu\, e_k^a(p_k^c)$. It can be shown that this switch and the "internal doubling" logic of NL2SOL are sufficient to negate the effects of the trust radius reduction when Δ_k is sufficiently small. However, if $\rho_k^c(p_k^c) < \eta_1$, an internal reduction cycle is executed which terminates with no assurance of a model switch. We therefore prefer the following.

For the instance where $\eta_2 > \rho_k^c(p_k^c) \geq \eta_1$, we consider the decision not to reduce the trust radius to be the superior strategy (particularly if the alternative model is to be tried first next step anyway). The decision to select p_k^c is also superior and is allowed by AS.5.3 and AS.5.5.

If $p_k^c(p_k^c) < \eta_1$, AS.5.7 allows p_k^c to be selected also as long as no reduction is subsequently done. We support our arguments against decreasing the trust radius by pointing out that model "a" did a significantly better job of prediction than model "c" by two different criteria, while model "c" picked a good direction but predicted the reduction poorly. The possibility that model "c" selected the better step by serendipity is strong enough to support not reducing Δ_k.

The following algorithm is an example of a two model system with no a priori preference between models. It is quite similar to NL2SOL, but with the changes noted above. The index "c" again denotes the currently preferred model, while "a" denotes the alternative.

Algorithm A.5.8 : Nonpreferential Model Switching

```
0) Let $\mu > 0$, $0 < \eta_1 \le \eta_2 \le \eta_3 < 1$, $0 < \gamma_1 < 1 \le \gamma_2 \le \gamma_3$, $\Delta_1 > 0$ and $x_1 \in \mathbb{R}^n$ be given.
   Set $\tau = 1$ and select a trial model $c$.

1) Repeat until convergence :

    2) Compute $p_k^c \in A(c, \Delta_k)$.

    3) If $\rho_k^c(p_k^c) < \eta_2$, then
            If $e_k^a(p_k^c) < \frac{1}{\mu} e_k^c(p_k^c)$, then
                Compute $p_k^a \in A(a, \Delta_k)$.
                If $\max \{ \rho_k^c(p_k^c), \rho_k^a(p_k^a) \} < \eta_1$ then go to 5).
                If $\rho_k^a(p_k^a) < \eta_2$ then set $\tau \in (0, \gamma_1]$.
                If $ared_k(p_k^c) \ge ared_k(p_k^a)$, then
                    Set $x_{k+1} = x_k + p_k^c$.
                    Go to 7).
                Else
                    Set $x_{k+1} = x_k + p_k^a$.
                    Switch $a$ and $c$.
                    Go to 7).
                End if.
            Else
                If $\rho_k^c(p_k^c) < \eta_1$ then go to 5).
                Set $x_{k+1} = x_k + p_k^c$.
                Set $\tau \in (0, \gamma_1]$.
                Go to 7).
                End if.
            End if.

    4) Set $x_{k+1} = x_k + p_k^c$.
        If $\rho_k^c(p_k^c) \ge \eta_3$, then set $\tau \in [\gamma_2, \gamma_3]$.
        Go to 7).

    5) Do internal reduction via Procedure A.5.4.

    6) Set $x_{k+1} = x_k + p_k^c$.

    7) Set $\Delta_{k+1} = \tau \Delta_k$.
    8) Set $\tau = 1$, update the models, and
       select a trial model for initial use in the next iteration.

9) End.
```

If both models are quadratic, Procedure A.5.4 in step 5) can be replaced by A.4.5.

Corollary 5.4 : *Given the assumptions of Theorem 5.3, Algorithm A.5.8 is FOSPC.*

Proof : Follows immediately from Theorem 5.3, Lemma 4.2, and examination of the algorithm. □

Another possible option for these algorithms is to include logic which explicitly uses knowledge about which models are quadratic with uniformly bounded curvature, and which models are only known to satisfy the UPD condition. For example, we can show that it is admissible to decrease the trust radius whenever $\rho_k^q(p_k^q) \le \eta_2$, or if $\rho_k^j(p_k^j) \le \eta_2$ and $\mu e_k^q(p_k^j) > e_k^j(p_k^j)$, where the model with index q is known to be quadratic with uniformly bounded curvature. However, the only use these changes might have would be to increase the efficiency of a system with several models by allowing Δ_k reduction without having to do as much testing between models. Since this task can be done better by simply increasing the value of μ, we find nothing to recommend such preferential logic.

5.10. Summary : Type Two Algorithms

Definition 5.2 : The general Algorithm A.4.3, when applied consistently with the specifications in AS.5.3, AS.5.5, AS.5.6, and AS.5.7, is called a **type two** algorithm.

This class of algorithms admits a very broad range of implementations of model switching and trust region logic. In particular, the major features of NL2SOL switching logic are admissible. There is no restriction on the number of models allowed, or on the method of choosing an initial trial model, and the algorithm is not required to use a priori information specifying which model has uniformly bounded curvature. There is no requirement forcing continuation of the switching logic tests once a trust radius reduction phase has started, and no requirement on how the trust radius is decreased as long as such decrease is allowed and as long as a satisfactory step is eventually found.

Furthermore, the class of algorithms based on Reduction_Test_1 can be implemented with high efficiency, even for a large number of models. These algorithms can be written to bias toward the use of a particular model if it is known to be inexpensive or suspected of being faster than the alternative, or they can treat the models solely on the basis of their performance as in NL2SOL.

Finally, only two requirements are imposed on the models. At least one model must be a quadratic in standard form with uniformly bounded curvature, and each of the models and associated procedure for computing a trial step must satisfy the UPD condition.

CHAPTER 6

Safe Algorithms using Minimal Restrictions on the Models

6.1. Introduction

In this chapter, we only assume that one model is a standard quadratic with uniformly bounded curvature. It is implicit in the definition of the previous algorithms that the number of models need not be constant from iteration to iteration. Thus any type two algorithm that *deletes* all models not known to satisfy the uniform predicted decrease condition is still first order stationary point convergent. Alternately, if all such models are *modified* so that the condition is assured, then any type two algorithm using these models is first order stationary point convergent. A type two algorithm that deletes or modifies models not known to satisfy the uniform predicted decrease condition is said to be a **type three algorithm.**

6.2. Enforcing the UPD condition by Direct Methods

One technique for ensuring that (5.1) holds for each p_k^i considered is to directly enforce such a condition. The expression

$$minred_k\,(\Delta_k) \equiv c_0 \,||\, g_k \,||\, \min\left\{ \Delta_k \,, \frac{||\, g_k \,||}{\xi_k} \right\} \tag{6.1}$$

can be used to decide whether to delete a model and its associated trial step from consideration at the current iteration. The parameter c_0 is set to a value in $(0,0.5)$. The variable $\xi_k > 0$ is an overestimate of the maximum curvature of f. It can be set to a constant value, such as 10^6. We then define

$$UPD\_Test_k^i(\Delta_k) = \begin{cases} \text{false} & \text{if } pred_k^i(p_k^i) < minred_k(\Delta_k) \\ \text{true} & \text{if } pred_k^i(p_k^i) \geq minred_k(\Delta_k) \end{cases} \quad . \tag{6.2}$$

6.2.1. A Globally Convergent Modification of NL2SOL

The following algorithm is a modification of Algorithm A.5.8 that is globally convergent under even milder assumptions on the models. Specifically, one of the two models used is assumed to be a quadratic in standard form, but no assumptions whatsoever are made concerning the alternate model. The model switching scheme is a very slight variation of that used by NL2SOL. The algorithm itself can be applied to NLS or general unconstrained optimization. For simplicity we have left out the internal doubling option. As in Algorithm A.5.8, the current model is denoted by the index c and the alternate is denoted by the index a. One of the models, denoted by the index q, is a quadratic in standard form.

Algorithm A.6.1 : A Globally Convergent Model Switching Algorithm

0) Let $\mu > 0$, $c_0 \in (0,0.5)$, $0 < \eta_1 \leq \eta_2 \leq \eta_3 < 1$, $0 < \gamma_1 < 1 \leq \gamma_2 \leq \gamma_3$, $j_c \in [1,\infty)$, $\Delta_1 > 0$ and $x_1 \in \mathbb{R}^n$ all be given.

```
Select a trial model with index $c$.

1) Repeat until convergence :

    2) Compute an overestimate $\xi_k$ to the curvature of the problem.

    3) Compute $p_k^c \in A(c, \Delta_k)$.

    4) If $c \neq q$ and $UPD\_Test_k^c(\Delta_k) =$ false, then
          Switch model preference by switching the indices $a$ and $c$.
          Compute $p_k^c \in A(c, \Delta_k)$.
          If $\rho_k^c(p_k^c) \leq \eta_1$ then go to 8).
       End if.

    5) If $\rho_k^c(p_k^c) \geq \eta_2$, then
          Set $x_{k+1} = x_k + p_k^c$.
          If $\rho_k^c(p_k^c) \geq \eta_3$ then set $\tau \in [\gamma_2, \gamma_3]$.
          Go to 9).
       End if.

    6) If $e_k^a(p_k^c) < \frac{1}{\mu} e_k^c(p_k^c)$, then
          Compute $p_k^a \in A(a, \Delta_k)$.
          If $a \neq q$ and $UPD\_Test_k^a(\Delta_k) =$ false, then
                If $\rho_k^c(p_k^c) < \eta_1$ then go to 8).
                Set $\tau \in (0, \gamma_1]$.
                Set $x_{k+1} = x_k + p_k^c$.
                Go to 9).
          End if.
          Set $\rho_k = \max\{\rho_k^c(p_k^c), \rho_k^a(p_k^a)\}$.
          If $\rho_k < \eta_1$ then go to 8).
          If $\rho_k < \eta_2$ then set $\tau \in (0, \gamma_1]$.
          If $ared_k(p_k^c) \geq ared_k(p_k^a)$, then
                Set $x_{k+1} = x_k + p_k^c$.
                Go to 9).
          Else,
                Set $x_{k+1} = x_k + p_k^a$.
                Switch $a$ and $c$.
                Go to 9).
          End if.
       End if.

    7) If $\rho_k^c(p_k^c) \geq \eta_1$, then
          Set $x_{k+1} = x_k + p_k^c$.
```

```
        Set $\tau \in (0, \gamma_1]$.
        Go to 9).
      End if.

   8) For $j=1,2,...$ do :
        Set $\Delta_k \in (0, \gamma_1 \Delta_k]$.
        Compute $p_k^c \in A(c, \Delta_k)$.
        If $c \neq q$ and $UPD\_Test_k^c(\Delta_k)=$ false, then
            Switch model preference by switching the indices $a$ and $c$.
            Compute $p_k^c \in A(c, \Delta_k)$.
        End if.
        If $\rho_k^c(p_k^c) \geq \eta_1$, then set $x_{k+1}=x_k+p_k^c$ and go to 9).
        If $c \neq q$ and $mod(j, j_c)=0$, then
            Switch model preference by switching the indices $a$ and $c$.
        End if.
      End for.

   9) If $\tau < 1$ then set $\Delta_{k+1}=\tau \| x_{k+1}-x_k \|$,
                  Else set $\Delta_{k+1}=\tau \Delta_k$.

   10) Set $\tau=1$, update the models, and select a trial model for the next iteration.

11) End.
```

Several features of this algorithm deserve comment. First, the parameter j_c is the integer introduced in Procedure A.5.4, and the function $mod(j, j_c)$ is the standard "modulus" operator that has value 0 if and only if $j=j_c$. The portions of the algorithm involving j_c are included to guarantee successful termination of the inner reduction loop 8) for general models and are are not essential if both models are quadratics. A suggested value for j_c is 5.

Second, this algoñithm is much less complicated than superficial appearances might indicate. Denote the alternative to model q with the index d. A.6.1 differs from A.5.8 only in that it deletes model d and step p_k^d if $pred_k^d(p_k^d)$ is not as large as expected.

Third, no restrictions are placed on how the new trial model is selected at the end of each iteration. This decision could be made by any of the tests for model selection discussed in Chapter 3. Another possibility is to continue to use the model which was successful at the current iteration. Yet another would be to initially try the same model at every iteration.

Using the theory presented in Chapter 5, we can establish conditions under which this algorithm is globally convergent. Consider the following specifications.

(A) One of the models, denoted by the index q, is a quadratic in standard form for which $\limsup_{k \to \infty} \| B_k^q \| < \infty$.

(B) If $p_k^q \in A(q, \Delta_k)$, then p_k^q is either an FCD step or an OLC step.

(C) For every i, $p_k^i \in A(i, \Delta_k)$ implies $\| p_k^i \| \leq (1 + \sigma_1) \Delta_k$ for some $\sigma_1 \in [0,1)$.

We then have the following result.

Theorem 6.1 : *For a given $x_1 \in \mathbb{R}^n$, assume that f satisfies the standard assumptions and that (A), (B), and (C) hold. If A.6.1 is applied to f starting from x_1 and $\{\xi_k\}$ is bounded, then the algorithm generates a sequence $\{x_k\}$ which is FOSPC.*

Proof : Any step p_k accepted by the algorithm satisfies $ared_k(p_k) > 0$, so $\{x_k\}$ remains within $L(f, x_1)$. Inspection of A.6.1 shows the existence of p_k unless step 8) is unable to find a step satisfying $p_k^i \in A(i, \Delta_k)$ and $\rho_k^i(p_k^i) \geq \eta_1$. But after $j_c < \infty$

executions of this loop, the model preference is switched to q. Condition (A) states that this model is a quadratic in standard form; and Lemma 4.2 establishes the existence at each iteration of a positive scalar δ for which $p_k^q \in A(q, \Delta_k)$ and $\Delta_k \le \delta$ implies $\rho_k^q(p_k^q) \ge \eta_1$. Since $\gamma_1 < 1$, the algorithm must eventually either accept a step with $\Delta_k > \delta$ or set $\Delta_k < \delta$. Hence A.6.1 successfully generates a step p_k at every iteration.

Next, conditions (A), (B), and Lemma 4.1 imply that the model with index q generates steps p_k^q satisfying the UPD condition. If the model with index d generates a step not satisfying $pred_k^d(p_k^d) \ge minred_k(\Delta_k)$, it is deleted from consideration. Since $\{\xi_k\}$ is bounded, each step p_k^i considered satisfies the UPD condition. Therefore this algorithm is an admissible variation of AS.5.7 and thus is FOSPC by Theorem 5.3.

□

6.2.2. Some Remarks on Direct Methods

Direct tests, such as used in the last example, are conceptually different from other tests we have presented. Notice that A.6.1 deletes models and associated steps *before* evaluating the actual function reduction. Direct tests should therefore be thought of as a way of limiting which models to include in I_k at a given iteration and should not be considered part of the model switching logic per se.

Moreover, great care should be taken in implementing these tests as they can be scale dependent. Suppose, for example, that a NLS algorithm using a secant model and a Gauss-Newton model sets $c_0 = 0.5$ and defines the sequence $\{\xi_k\}$ by

$\xi_k = \| J(x_k)^t J(x_k) \|$. If $S(x)$ is positive definite, $\| H(x) \|$ may typically be much larger than $\| J(x)^t J(x) \|$, so the secant model may be deleted even if it does a much better job of predicting the actual function reduction than the Gauss-Newton model. Even more striking is the case of using the identity matrix for B_k^g and using a secant method to generate the other model Hessian. A naive implementation might set $c_0 = 0.5$ and $\xi_k = \| I \| = 1$. Again, the secant model may be frequently rejected in favor of the model using $B_k^g = I$ even if it does a much better job of predicting the actual function reduction.

To avoid spuriously deleting valid models, c_0 should be set to a very small value. We suggest $c_0 = 0.001$. As previously stated, the sequence $\{\xi_k\}$ should be an *overestimate* to the curvature of the problem.

On the other hand, direct tests have several advantages. They are simple and inexpensive to implement. They can be applied to *any* models whether quadratic or not. Also, use of $minred_k$ to delete models is much better than, say, deleting models which violate $\| B_k^i \| > \xi_k$.

6.2.3. Estimating an Upper Bound for the Curvature of a Function

Direct tests are less sensitive to the scale of the problem if the curvature estimates used to define ξ_k are determined from the behavior of the function, rather than from the curvature of one of the models. Such estimates can be produced as follows.

Consider a secant method for "updating" a model Hessian B_k. Such a method generates a new model Hessian B_{k+1} satisfying the secant equation

$$B_{k+1} p_k = y_k \tag{6.3}$$

for some $y_k \in \mathbb{R}^n$. Define $\tilde{H}_{k+1} \equiv \int_0^1 H(x_k + \lambda p_k)\, d\lambda$ and recall that Lemma 2.4 established

$$\int_0^1 H(x_k + \lambda p_k) p_k \, d\lambda = g_{k+1} - g_k \ . \tag{6.4}$$

Thus, if y_k is defined to be $g_{k+1} - g_k$, a matrix B_{k+1} satisfying the secant equation has roughly the same curvature as $\tilde{H}_{k+1}$ in the direction p_k. For nonzero p_k, this suggests the curvature estimate

$$\begin{aligned} \xi_{k+1} &= \max\left\{ \xi_{k-1}, \left| \frac{p_k^t B_{k+1} p_k}{p_k^t p_k} \right| \right\} \\ &= \max\left\{ \xi_{k-1}, \frac{| p_k^t y_k |}{p_k^t p_k} \right\} . \end{aligned} \tag{6.5}$$

This estimator is bounded if there exist $\beta_4 > 0$ such that

$$| p_k^t y_k | \le \beta_4 \| p_k \|^2 \ . \tag{6.6}$$

For the standard definition of y_k, this is easily established.

Lemma 6.2 : *For a given $x_1 \in \mathbb{R}^n$, assume that f satisfies the standard assumptions.*

For any x_{k+1} and x_k in $L(f, x_1)$, let y_k be defined as

$$y_k = g_{k+1} - g_k \ . \tag{6.7}$$

Then

$$| y_k^t p_k | \leq \beta_1 \| p_k \|^2 \ . \qquad (6.8)$$

Proof : From the Cauchy-Schwartz inequality, $| y_k^t p_k | \leq \| p_k \| \ \| y_k \|$. Lemma 2.5 and (SA3) imply $\| g_{k+1} - g_k \| \leq \beta_1 \| p_k \|$, which leads immediately to the result.

□

Other formula are sometimes used to define y_k. Alternate definitions of y_k have been created[1] for NLS problems in order to take advantage of the problem structure. For most of these alternate definitions, mild conditions on r are sufficient to establish that $\dfrac{| y_k^t p_k |}{\| p_k \|^2}$ is bounded. For example, DGW [1981] make the definition

$$y_k = J_{k+1}^t r_{k+1} - J_k^t r_{k+1} + J_{k+1}^t J_{k+1} p_k \ . \qquad (6.9)$$

Consider the following conditions.

(i) The sequence $\{ x_k \}$ satisfies the descent property with respect to $f(x) = \frac{1}{2} \| r(x) \|_2^2$.

(ii) There exists $\beta_J < \infty$ such that for any $x \in L(f, x_1)$, $\| J(x) \| \leq \beta_J$.

(iii) $J(x)$ is Lipschitz continuous over $L(f, x_1)$ so that for every $x, y \in L(f, x_1)$ we have $\| J(x) - J(y) \| \leq \beta_{lip} \| x - y \|$.

Lemma 6.3 : *Let y_k be defined by (6.9) and suppose that (i), (ii) and (iii) above are true. Then there exists $\beta_4 < \infty$ such that*

[1] See, for example, DGW [1981], Dennis, Sheng, Vu [1985], Betts [1976], Dennis [1973], and Al-Baali, Fletcher [1985].

$$| y_k^t p_k | \leq \beta_4 || p_k ||^2 \ . \tag{6.10}$$

Proof : First, we have

$$\begin{aligned} y_k^t p_k &= p_k^t (J_{k+1}^t - J_k^t) r_{k+1} + p_k^t J_{k+1}^t J_{k+1} p_k \\ &= || J_{k+1} p_k ||^2 + p_k^t \left[J(x_k + p_k)^t - J(x_k)^t \right] r_{k+1} \ . \end{aligned} \tag{6.11}$$

From the Cauchy-Schwartz inequality

$$| p_k^t y_k | \leq || J_{k+1} ||^2 || p_k ||^2 + || p_k || \ || \left[J(x_k + p_k)^t - J(x_k)^t \right] r_{k+1} ||$$

so that applying (i), (ii) and (iii) gives

$$\begin{aligned} | p_k^t y_k | &\leq \left[\beta_J{}^2 + \beta_{lip} \ || r_{k+1} || \right] || p_k ||^2 \\ &\leq \left[\beta_J{}^2 + \beta_{lip} \sqrt{2f(x_k)} \right] || p_k ||^2 \\ &\leq \left[\beta_J{}^2 + \beta_{lip} \sqrt{2f(x_1)} \right] || p_k ||^2 \ . \end{aligned} \tag{6.12}$$

Therefore defining $\beta_4 = \beta_J{}^2 + \beta_{lip} \sqrt{2f(x_1)}$ completes the proof. □

We make use of the following procedure to compute ξ_k.

Procedure A.6.2 : Computing a bounded overestimate ξ_k for the curvature of a function

```
If $k = 1$ then set $\xi_1 = 10^6$.
If $k > 1$ and $\| p_{k-1} \| \neq 0$, then
    Set $\xi_k = \max \{ \xi_{k-1}, 100 \times \frac{y_{k-1}^t p_{k-1}}{p_{k-1}^t p_{k-1}} \}$.
End if.
```

Implementation of this procedure under the assumptions of Lemma 6.2 or Lemma 6.3 generates a bounded sequence $\{\xi_k\}$ which can be used in A.6.1 or other algorithms using direct tests to enforce UPD. An alternative is to use a fixed curvature estimator, such as $\xi_k = 10^6$.

6.3. The UPD Condition and Quadratic Models

In Chapter 4, we considered quadratics in standard form and assumed that steps were computed using FCD or OLC procedures. Lemma 4.1 established that

$$\limsup_{k \to \infty} \| B_k^i \| < \infty \tag{6.13}$$

is sufficient to imply the UPD condition for the model with index i. However, much weaker assumptions can be used to establish the UPD condition. These weaker conditions differ not only for different categories of models but also for different step generation procedures.

6.3.1. Sufficient Conditions for UPD with Quadratic Models

Before proving our theorem, we review the following results concerning quadratics in one variable and their extrema. Consider a nonzero vector $w \in \mathbb{R}^n$. For a quadratic model in standard form at iteration k with model Hessian B_k^i we have

$$pred_k^i(\alpha w) = -\alpha g_k^t w - \frac{1}{2}\alpha^2 w^t B_k^i w \quad . \tag{6.14}$$

Now, if $w^t B_k^i w > 0$, we have that

$$\underset{\alpha}{\text{maximize}}\, pred_k^i(\alpha w) \tag{6.15}$$

is solved by $\alpha^* = -\dfrac{g_k^t w}{w^t B_k^i w}$, with

$$pred_k(\alpha^* w) = \frac{1}{2}\frac{(g_k^t w)^2}{w^t B_k^i w}. \tag{6.16}$$

Furthermore, for $\alpha \in [0, \alpha^*]$

$$-\frac{\alpha}{2} g_k^t w \leq pred_k(\alpha w) \leq -\alpha g_k^t w \quad . \tag{6.17}$$

If $w^t B_k^i w \leq 0$, we have that

$$pred_k(\alpha w) \geq -\alpha g_k^t w. \tag{6.18}$$

for all α. We then have 3 possible cases in solving

$$\underset{\alpha}{\text{maximize}}\, pred_k^i(\alpha w) \quad . \tag{6.19}$$

$$s/t \ \| \alpha w \| \leq \Delta_k$$

(1) If $w^t B_k^i w > 0$ and $\| \alpha^* w \| \leq \Delta_k$, then (6.19) is solved by α^* as before, with maximum

$$pred_k\,(\alpha^* w) = \frac{1}{2}\frac{(g_k^t w)^2}{w^t B_k^i w} \quad . \tag{6.20}$$

(2) If $w^t B_k^i w > 0$ and $\| \alpha^* w \| > \Delta_k$, then (6.19) is solved by

$$\alpha^{**} = -sign\,(g_k^t w)\,\Delta_k / \| w \| \tag{6.21}$$

with maximum

$$pred_k^i\,(\alpha^{**} w) = sign\,(g_k^t w)\frac{g_k^t w}{\| w \|}\Delta_k - \frac{1}{2}\Delta_k^2 \frac{w^t B_k^i w}{\| w \|^2} \quad . \tag{6.22}$$

Furthermore

$$\frac{1}{2}\Delta_k \frac{| g_k^t w |}{\| w \|} \le pred_k^i\,(\alpha^{**} w) \le \Delta_k \frac{| g_k^t w |}{\| w \|} \ . \tag{6.23}$$

(3) If $w^t B_k^i w \le 0$, then (6.19) may have either one or two local solutions corresponding to $\alpha^{***} = \pm \frac{\Delta_k}{\| w \|}$. If $g_k^t w = 0$, two maxima exist with

$$pred_k^i\,(\alpha^{***} w) = -\frac{1}{2}\left[\frac{\Delta_k}{\| w \|}\right]^2 w^t B_k^i w \quad . \tag{6.24}$$

If not, the global maximizer is unique so that

$$\alpha^{***} = -sign\,(g_k^t w)\frac{\Delta_k}{\| w \|} \tag{6.25}$$

and

$$pred_k^i\,(\alpha^{***} w) = | g_k^t w | \frac{\Delta_k}{\| w \|} - \frac{1}{2}\left[\frac{\Delta_k}{\| w \|}\right]^2 w^t B_k^i w \ . \tag{6.26}$$

In either event, the maximum is given by (6.26). Furthermore,

$$\Delta_k \frac{| g_k^t w |}{\| w \|} \le pred_k^i\,(\alpha^{***} w) \ . \tag{6.27}$$

The following theorem applies to quadratic models that use FCD or OLC procedures to generate trial steps.

Theorem 6.4 : *Consider a quadratic model in standard form with model Hessian B_k^i and an associated step p_k^i.*

(a) If p_k^i is an FCD step with constant $c_1 > 0$ and

$$g_k^t B_k^i g_k \le \beta_3 g_k^t g_k \tag{6.28}$$

for some $\beta_3 > 0$, then

$$pred_k^i(p_k^i) \ge \frac{1}{2} c_1 \| g_k \| \min \left\{ \Delta_k , \frac{\| g_k \|}{\beta_3} \right\} . \tag{6.29}$$

(b) Let p_k^i be an OLC step with constant $c_2 > 0$ and let $w_k \in \mathbb{R}^n$ be a nonzero vector satisfying

$$w_k^t B_k^i w_k \le \beta_4 w_k^t w_k \tag{6.30}$$

and

$$| g_k^t w_k | \ge \varepsilon_4 \| g_k \| \, \| w_k \| \tag{6.31}$$

for some $\beta_4 > 0$, $\varepsilon_4 > 0$. Then

$$pred_k^i(p_k^i) \ge \frac{1}{2} c_2 \varepsilon_4 \| g_k \| \min \left\{ \Delta_k , \frac{\varepsilon_4 \| g_k \|}{\beta_4} \right\} . \tag{6.32}$$

Proof :

(a) Let p_k^i be an FCD step so that

$$pred_k^i\,(p_k^i\,) \geq c_1 \max_{\substack{\alpha \\ s/t\ \|\alpha g_k\| \leq \Delta_k}} pred_k^i\,(\alpha\, g_k\,) \quad . \tag{6.33}$$

Now let w be replaced by g_k in (6.19), (6.20), (6.23) and (6.27) so that (6.28) implies

$$\begin{aligned} pred_k^i\,(p_k^i\,) &\geq c_1 \min\left\{ \frac{1}{2}\,\frac{\|g_k\|^4}{g_k^t B_k^i g_k}\,,\, \frac{1}{2}\Delta_k\,\frac{g_k^t g_k}{\|g_k\|} \right\} \\ &\geq \frac{1}{2} c_1 \|g_k\| \min\left\{ \Delta_k\,,\, \frac{\|g_k\|}{\beta_3} \right\} \quad , \end{aligned} \tag{6.34}$$

which establishes the part (a) of the theorem.

(b) Let p_k^i satisfy OLC condition so that

$$\begin{aligned} pred_k^i\,(p_k^i\,) &\geq c_2 \max_{\substack{p \\ s/t\ \|p\| \leq \Delta_k}} pred_k^i\,(p\,) \\ &\geq c_2 \max_{\substack{\alpha \\ s/t\ \|\alpha w_k\| \leq \Delta_k}} pred_k^i\,(\alpha\, w_k\,)\,. \end{aligned} \tag{6.35}$$

From (6.19),(6.20), (6.23), (6.27), (6.30) and (6.31) we obtain

$$\begin{aligned} pred_k^i(p_k^i) &\geq c_2 \min\left\{ \frac{1}{2} \frac{|\, g_k^t w_k \,|^2}{w_k^t B_k^i w_k}, \frac{1}{2} \Delta_k \frac{|\, g_k^t w_k \,|^2}{\| w_k \|} \right\} \\ &\geq \frac{1}{2} c_2 \min\left\{ \frac{\varepsilon_4^2 \| g_k \|^2 \| w_k \|^2}{w_k^t B_k^i w_k}, \Delta_k \frac{\varepsilon_4 \| g_k \| \; \| w_k \|}{\| w_k \|} \right\} \\ &= \frac{c_2 \varepsilon_4}{2} \| g_k \| \min\left\{ \Delta_k , \frac{\varepsilon_4 \| g_k \| \; \| w_k \|^2}{w_k^t B_k^i w_k} \right\} \\ &\geq \frac{c_2 \varepsilon_4}{2} \| g_k \| \min\left\{ \Delta_k , \frac{\varepsilon_4}{\beta_4} \| g_k \| \right\} \end{aligned} \tag{6.36}$$

which establishes the part (b) of the theorem. □

6.3.2. Directionally Bounded Curvature

Consider a sequence of nonzero vectors $\{v_k\}$ in $\mathbb{R}^n$. A sequence of quadratic models in standard form is said to be **directionally** bounded in curvature with respect to $\{v_k\}$ if there exists $\beta_3 > 0$ such that

$$v_k^t B_k^i v_k \leq \beta_3 \, v_k^t v_k \tag{6.37}$$

for all k. If there exist $\varepsilon_4 > 0$ and $\beta_4 > 0$ such that for any k there exists a nonzero $w_k \in \mathbb{R}^n$ satisfying

$$w_k^t B_k^i w_k \leq \beta_4 \, w_k^t w_k \tag{6.38}$$

and

$$|\, v_k^t w_k \,| \geq \varepsilon_4 \| v_k \| \; \| w_k \| \; , \tag{6.39}$$

then the sequence of models is said to be **quasi-directionally** bounded in curvature with respect to $\{v_k\}$.

Corollary 6.5 : *Consider a sequence of quadratic models in standard form with an associated procedure for computing steps.*

(a) If the procedure computes FCD steps and the sequence of models is of directionally bounded curvature with respect to $\{g_k\}$, *then the UPD condition holds.*

(b) If the procedure computes OLC steps and the sequence of models is of quasi-directionally bounded curvature with respect to $\{g_k\}$, *then the UPD condition holds.*

Proof : To establish (a), let the constant c_3 in (5.1) be defined as $c_3 \equiv \frac{1}{2} c_1$. Then (a) follows immediately from Theorem 6.4. To establish (b), let the constants in (5.1) be defined as $c_3 \equiv \frac{1}{2} c_2 \varepsilon_4$ and $\beta_3 = \frac{\beta_4}{\varepsilon_4}$. Then (b) follows immediately from Theorem 6.4. □

6.3.3. Relation Between Directionally Bounded Curvature and Uniformly Bounded Curvature

Consider a sequence of models which is directionally bounded in curvature with respect to $\{g_k\}$ so that

$$g_k^t B_k^i g_k \le \beta_3 \, g_k^t g_k \tag{6.40}$$

for some β_3. We immediately see that (6.40) is weaker than (6.13); and hence Lemma 4.1 is a special case of Theorem 6.4.

6.3.4. Quasi-directionally Bounded Curvature and Secant Methods

A sequence of models is quasi-directionally bounded in curvature with respect to $\{g_k\}$ only if there exist $\varepsilon_4 > 0$ and $\beta_4 > 0$ for which

$$w_k^t B_k^i w_k \le \beta_4 \, w_k^t w_k \tag{6.41}$$

and

$$| \; g_k^t w_k \; | \ge \varepsilon_4 \, \| g_k \| \; \| w_k \| \tag{6.42}$$

for some nonzero $w_k \in \mathbb{R}^n$ at every iteration. We immediately see that (6.41) and (6.42) are a strict relaxation of (6.40).

Next, consider a secant method producing model Hessians which satisfy $B_k^i p_{k-1} = y_{k-1}$. Substituting p_{k-1} for w_k in (6.41) and (6.42) gives the expressions

$$p_{k-1}^t B_k^i p_{k-1} \le \beta_4 \, p_{k-1}^t p_{k-1} \tag{6.43}$$

and

$$| \; g_k^t p_{k-1} \; | \ge \varepsilon_4 \, \| g_k \| \; \| p_{k-1} \| \quad . \tag{6.44}$$

As noted earlier, most definitions of y_k ensure (under mild assumptions) that $\dfrac{| \, p_k^t y_k \, |}{p_k^t p_k}$ is bounded. If so, there exists $\beta_4 < \infty$ for which

$$\begin{aligned}\beta_4 p_{k-1}^t p_{k-1} &\geq p_{k-1}^t y_{k-1} \\ &= p_{k-1}^t B_k^i p_{k-1} \ .\end{aligned} \tag{6.45}$$

Hence by Theorem 6.4, if (a) $\dfrac{| p_k^t y_k |}{p_k^t p_k}$ is bounded by β_4, (b) B_k^i satisfies the secant equation $B_k^i p_{k-1} = y_{k-1}$ and (c) p_k^i is computed by an OLC procedure, then (6.44) is a sufficient condition to imply the UPD condition.

6.3.5. Alternatives to UPD_Test

One possible use of Theorem 6.4 is to provide alternatives to the use of *minred*$_k$ for enforcing the UPD condition. Rather than using (6.2) to identify questionable models, we can define

$$UPD\_Test\_FCD_k^i = \begin{cases} \text{false} & \text{if } g_k^t B_k^i g_k > \xi_k \| g_k \|^2 \\ \\ \text{true} & \text{if } g_k^t B_k^i g_k \leq \xi_k \| g_k \|^2 \ . \end{cases} \tag{6.46}$$

Unlike *UPD_Test*, this test need only be performed at the start of an iteration rather than every time a step is computed. Furthermore, it is less expensive since it can be applied before computation of a step. For secant methods using OLC procedures to compute trial steps, we can also use the test

$$UPD\_Test\_OLC_k^i = \begin{cases} \text{false} & \begin{array}{l}\text{if } g_k^t B_k^i g_k > \xi_k \| g_k \|^2 \\ \quad \text{and } | g_k^t p_{k-1} | < \varepsilon_0 \| g_k \| \ \| p_{k-1} \| \end{array} \\ \\ \text{true} & \begin{array}{l}\text{if } g_k^t B_k^i g_k \leq \xi_k \| g_k \|^2 \\ \quad \text{or } | g_k^t p_{k-1} | \geq \varepsilon_0 \| g_k \| \ \| p_{k-1} \| \end{array} \end{cases} \tag{6.47}$$

for some $\varepsilon_0 > 0$. A suggested value is $\varepsilon_0 = 10^{-6}$.

6.3.6. Modifying Quadratics to be Directionally Bounded

Theorem 6.4 shows that the UPD condition is satisfied as long as the curvature of the standard quadratic model is bounded in the gradient direction. This suggests an alternative to simply deleting models not satisfying $UPD\_Test_k^i = \text{true}$. These models can instead be modified in a subspace containing g_k so that the curvature in the gradient direction is *corrected.*

Several procedures are available to force the curvature in a given direction to take on a specific value. One of the simplest is as follows. Let $U(B,p,y): \mathbb{R}^{n \times n} \times \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ represent a secant update satisfying $U(B,p,y)p = y$.

Procedure A.6.3 : Simple Procedure for Correcting a Model Hessian

```
If $UPD\_Test\_FCD_k^i$ = false, then
    Set $\beta = \xi_k$.
    Set $p^\varepsilon = g_k$.
    Set $y^\varepsilon = \beta g_k$.
    Compute $B = U(B_k^i, p^\varepsilon, y^\varepsilon)$.
    Set $B_k^i = B$.
End if.
```

This procedure only modifies B_k^i if $g_k^t B_k^i g_k > \xi_k \, g_k^t g_k$. After modification, B_k^i will satisfy $B_k^i p^\varepsilon = y^\varepsilon$ and $g_k^t B_k^i g_k = \xi_k \, g_k^t g_k$. For a typical secant update, the model is only changed in a two dimensional subspace. If desired, an update can be used which also forces B_k^i to continue to satisfy the secant equation from the last iteration:

$$B_k^i p_{k-1} = y_{k-1} \quad . \tag{6.48}$$

The procedure can be improved by setting β to a finite difference approximation of $\dfrac{g_k^t H(x_k) g_k}{g_k^t g_k}$. One such approximation is

$$\beta = \frac{2}{\varepsilon^2} \frac{f(x_k + \varepsilon g_k) - f(x_k) - \varepsilon \| g_k \|^2}{\| g_k \|^2} \quad . \tag{6.49}$$

Techniques for determining an appropriate value for ε can be found in Section 5.4 of

Dennis and Schnabel [1983]. A central difference approximation could also be used, but would require an extra function evaluation.

Procedure A.6.4 : Another Procedure for Correcting a Model Hessian

```
If $UPD\_Test\_FCD_k^i$ = false, then
    Set $\beta = \frac{2}{\varepsilon^2} \frac{f(x_k + \varepsilon g_k) - f(x_k) - \varepsilon \| g_k \|^2}{\| g_k \|^2}$ for an appropriately selected $\varepsilon$.
    Set $p^\varepsilon = g_k$.
    Set $y^\varepsilon = \beta g_k$.
    Compute $B = U(B_k^i, p^\varepsilon, y^\varepsilon)$.
    Set $B_k^i = B$.
End if.
```

After modification, B_k^i will satisfy $B_k^i p^\varepsilon = y^\varepsilon$ and hence $g_k^t B_k^i g_k = \frac{2}{\varepsilon^2} (f(x_k + \varepsilon g_k) - f(x_k) - \varepsilon \| g_k \|^2)$.

Notice that ξ_k is only used in determining whether modification of the matrix is deemed necessary. The modification itself does not depend on any arbitrary constants. Hence, even if ξ_k is fixed at an arbitrary value, Procedure A.6.4 is essentially scale invariant.

Although A.6.4 is a large improvement to AS.6.3, it still has one undesirable feature. Specifically, the algorithm alway modifies B_k^i so that g_k is an eigenvector. The following procedure does not share this feature.

Procedure A.6.5 : The Recommended Procedure for Correcting a Model Hessian

```
If $UPD\_Test\_FCD_k^i$ = false, then
    For an appropriately selected $\varepsilon$,
        Set $p^\varepsilon = \varepsilon g_k$.
        Set $y^\varepsilon = g(x_k + p^\varepsilon) - g_k$.
        Compute $B = U(B_k^i, p^\varepsilon, y^\varepsilon)$.
        Set $B_k^i = B$.
End if.
```

After modification, B_k^i will satisfy

$$B_k^i g_k = \frac{1}{\varepsilon} \left(g(x_k + \varepsilon g_k) - g_k \right) . \tag{6.50}$$

Lemma 6.6 : *Let f satisfy the standard assumptions and let x_k be in $\tilde{L}(f, x_1)$. Let ε be any nonzero scalar satisfying $x_k + \varepsilon g_k \in \tilde{L}(f, x_1)$.*

If (6.50) is satisfied, then $g_k^t B_k^i g_k \leq \beta_1 \| g_k \|^2$.

Proof : Application of Lemma 2.5 and the standard assumptions on f yield

$$\begin{aligned} \| B_k^i g_k \| &\leq \frac{1}{\varepsilon} \| g(x_k + \varepsilon g_k) - g_k \| \\ &\leq \frac{1}{\varepsilon} \beta_1 \| \varepsilon g_k \| = \beta_1 \| g_k \| \quad . \end{aligned} \tag{6.51}$$

By the Cauchy-Schwartz inequality

$$g_k^t B_k^i g_k \leq \| g_k \| \, \| B_k^i g_k \| \leq \beta_1 \| g_k \|^2 \ , \tag{6.52}$$

which completes the proof. □

If OLC steps are being computed for a given secant model, *UPD_Test_FCD* can be replaced by *UPD_Test_OLC* in any of last three procedures provided $\dfrac{| p_k^t y_k |}{p_k^t p_k}$ is bounded.

6.4. Summary : Type Three Algorithms

The methods of this chapter make it possible to use any model in an algorithm and retain FOSPC under the standard assumptions on the function. These methods are computationally efficient and can be implemented in a relatively scale independent way.

Three tests are presented for identification of questionable models. *UPD_Test* can be used with any model, while *UPD_Test_FCD* and *UPD_Test_OLC* apply to quadratics in standard form with associated FCD or OLC procedures for computing trial steps. These tests are performed directly on the models and thus do not involve any comparison of the model with the function.

Once a questionable model is identified, it can either be deleted from the set of models under consideration or be modified to satisfy the UPD condition. Deletion is the simplest procedure, but should be avoided if possible. For quadratic models, we strongly recommend modification by Procedure A.6.5.

Chapters 5 and 6 provide several theoretically justified ways of deciding when to "refresh" or reset a secant update to the true Hessian. First, a reset should be done if one of the tests in this chapter fail for the secant Hessian (e.g., if $g_k^t B_k g_k \geq \xi_k \, g_k^t g_k$). Furthermore, a reset should be done if a step computed with the secant model yields $\rho_k^1(p_k^1) < \eta_2$ (so that a trust region decrease is called for) unless the Newton model

$$pred_k^2(p) = -g_k^t p - \frac{1}{2} p^t H_k p \tag{6.53}$$

yields $\mu \, e_k^2(p_k^1) \geq e_k^1(p_k^1)$. Notice that it is *not* necessary to evaluate $H(x_k)$ to obtain $pred_k^2(p_k^1)$. We can instead determine an approximation to $pred_k^2(p_k^1)$ using finite differences. A simple way to do this is to use

$$\begin{aligned} pred_k^2(p_k^1) &= -g_k^t p_k^1 - \frac{1}{2}(p_k^1)^t H_k (p_k^1) \\ &\approx -g_k^t p_k^1 - \frac{1}{2}(p_k^1)^t y_k^\varepsilon \end{aligned} \tag{6.54}$$

where

$$y_k^\varepsilon = \frac{1}{\varepsilon}(g(x_k + \varepsilon p_k^1) - g_k)$$

and ε is a small positive number selected as in Section 5.4 of Dennis and Schnabel [1983]. A second order finite difference equation which only requires one extra function evaluation could also be used.

A simpler alternative to this finite difference procedure is to use the techniques presented in Chapter 5 associated with AS.5.6. A test to guarantee the UPD condition (such as : ``Refresh if $g_k^t B_k g_k \geq \xi_k \, g_k^t g_k$'') is still required.

CHAPTER 7

Concluding Remarks

7.1. Some Results Which Are Not Applicable To These Algorithms

We include the following examples to show properties that *are not* necessarily true for the algorithms we have presented.

7.1.1. An Example Concerning Local Convergence Rates.

The algorithms of Chapters 5 through 6 rely on the comparison of predicted function reduction with actual function reduction to distinguish between models. It might be speculated that such methods have local convergence rates of the same order as would be obtained by a single model algorithm that selects the "best" model at each iteration. However, consider the following problem. Let the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be defined so that

$$f(x) = \frac{1}{2} x^t x \quad , \quad g(x) = x \quad , \text{and} \quad H(x) = I \quad . \tag{7.1}$$

Thus f has minimizer $x^* = 0$ with $f(x^*) = 0$.

For all $k > 0$, define two possible model Hessians by

$$B_k^1 = I \text{ and } B_k^2 = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix} \quad . \tag{7.2}$$

Notice that B_k^1 is exact and that B_k^2 is positive definite with $(B_k^2)^{-1} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 3 \end{bmatrix}$.

Suppose that we use these two models to implement an algorithm such as A.6.1 and suppose that the currently preferred model is always selected as the initial trial model for the next iteration unless the alternate model does a better job of predicting the actual function reduction. Further suppose that the algorithm is started with $x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\bar{\Delta}_1 = 2$ and with B_1^2 as the initial trial model. It follows that

$$p_1^2 = -(B_1^2)^{-1} g_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad , \tag{7.3}$$

$$pred_1^2\,(p_1^2) = -g_1^t p_1^2 - \frac{1}{2}(p_1^2)^t\, B_1^2\,(p_1^2) = \frac{1}{2} \quad , \tag{7.4}$$

and

$$ared_1(p_1^2) = \frac{1}{2} \quad , \tag{7.5}$$

so that $\rho_1^2(p_1^2) = 1$. Thus the trial step is accepted, and the next iterate is $x_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

With no reason to change models the algorithm retains model two as the initial trial model in the next iteration. Furthermore, no change in trust radius is required. It follows that the next iteration generates

$$p_2^2 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} , \quad \rho_2^2(p_2^2) = 1 \quad . \tag{7.6}$$

As with the previous iteration, the trial step is acceptable, so that the third iterate is

$$x_3 = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} . \tag{7.7}$$

Again, no trust radius reduction or model switch is required before the next iteration.

Clearly, the iterates generated by this algorithm satisfy $x_k = \frac{1}{2} x_{k-2}$ for $k > 2$, and for every $k \geq 1$ we have $\| x_{k+1} - x^* \| = \frac{\sqrt{2}}{2} \| x_k - x^* \|$. Despite this, we have $\rho_k^2 (p_k^2) = 1$ and thus $ared_k(p_k^2) = pred_k^2(p_k^2)$ at *every* iteration.

We see that tests based on comparing actual reduction to predicted reduction are capable of retaining a model that yields slow linear convergence even though an alternate model would converge in one iteration. Furthermore, this example does not depend on any particular trust region or line search method since each step is a full quasi-Newton step with the trust region constraint inactive. The example can thus be applied to most formulations of each of the basic types of algorithms discussed in Chapter 3 since most are essentially equivalent whenever the trust region constraint is not binding.

7.1.2. An Example Concerning Convergence to a Saddle Point.

The single model trust region algorithms of SSB [1985] and Moré, Sorensen [1982, 1983] make use of negative curvature in solving the trust region subproblem (TRS) in such a way that convergence to saddle points is inhibited. Specifically, these algorithms have the property of second order stationary point convergence[1] when implemented with the exact Hessian.

Suppose that a type two or type three algorithm using two quadratic models is implemented with the exact Hessian as one of the model Hessians. It might be speculated that such an algorithm also shares the property of second order stationary point convergence.

However, consider the following problem for $x = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}$. We define the function $f : \mathbb{R}^3 \to \mathbb{R}$ as

$$f(x) = \frac{1}{2}\left[(\xi_1)^2 + (\xi_2)^2 - (\xi_3)^2 \right] \tag{7.8}$$

so that the gradient and Hessian are expressed by

$$g(x) = \begin{bmatrix} \xi_1 \\ \xi_2 \\ -\xi_3 \end{bmatrix} \quad , \quad H(x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad . \tag{7.9}$$

This function is indefinite with a saddle point at $x^* = 0$.

Define two possible model Hessians by

$$B_k^1 = H(x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{7.10}$$

[1] An algorithm is second order stationary point convergent if (a) it generates iterates $\{x_k\}$ that are FOSPC, and (b) the matrix $H(x^*)$ is positive semidefinite at any point x^* for which $x_k \to x^*$.

and

$$B_k^2 = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{7.11}$$

Suppose that the algorithm described in Section 7.1.1 (which retains the currently preferred model unless the alternate is significantly better) is applied to this problem. It is easily verified that if the algorithm is started with $x_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ and model two as the initial trial model, then the third component of x_k is identically zero for all $k \geq 1$. Furthermore, $\rho_k^2(p_k^2) = 1$ for all $k \geq 1$ as in the last example. Consequently, model one (the exact Hessian) is never selected as a trial model and the iterates converge to the saddle point $x = 0$.

7.2. Work in Progress

We conclude this thesis with a brief discussion of two of the more promising areas currently being investigated.

7.2.1. Secant Methods and Directional Boundedness

In Chapter 6, we showed that quadratic models that are uniformly directionally bounded will satisfy the UPD condition. We speculate that UPD need not be directly enforced for secant methods. Currently, attempts are being made both to (a) relax the requirement of UPD in our theory in such a way as to admit secant updates, and (b) show that existing secant updates are uniformly directionally bounded. Several encouraging results[2] have been established, but at this time the question remains open.

7.2.2. Achieving Nearly Optimal Local Convergence Rates

If the convergence rate associated with each model is known in advance, achieving the optimal rate for a multi-model algorithm is simply a matter of, at each iteration, initially selecting the model associated with the highest rate. For example, most of the algorithms of Al-Baali and Fletcher [1985] are biased toward selecting the secant approximation. This is consistent with the assumption that the secant model is associated with faster convergence than the Gauss-Newton model.

Unfortunately, one of the greatest potential uses of a multi-model algorithm is to efficiently solve problems of many different categories with one algorithm. Consider the NLS problem. As previously mentioned, a secant model can be associated with either faster or slower convergence rates than the Gauss-Newton model depending on whether the problem is zero residual, small residual, or large residual. Other examples are easily found where a priori knowledge is not available about ''which model is right.'' The lack of a nonpreferential multi-model algorithm that guarantees fast local convergence is thus unacceptable.

The most promising approach for designing such a method seems to be as follows. Consider an algorithm such as A.3.1 . We refer to an iteration that

[2] For example, it can be shown that if a subsequence of the models satisfy the UPD condition, versions of type one and type two algorithms exist that are WFOSPC.

computes a trial step for *every* model as a **greedy step**. Such an iteration is usually[3] capable of selecting between trial steps computed using an exact Hessian and trial steps computed using a constant Hessian approximation. The ability to make this distinction is crucial to achieving fast local convergence rates and to extending the Moré and Sorensen [1982, 1983] second order stationary point convergence results to multi-model algorithms. Algorithm A.3.1 performs a greedy step at each iteration and selects the model yielding the largest function reduction. Although this algorithm is too expensive to be useful for general problems, a similar method that only requires a greedy step at infrequent intervals should not be prohibitively expensive. Current research is centered on finding inexpensive algorithms that are seldom required to execute a greedy step yet are assured of converging with a nearly optimal rate.

[3] For the cases in which the values of $ared_k$ at different trial steps are ''indistinguishable,'' it can be shown that any of the trial steps can be selected without significantly effecting the convergence rates.

BIBLIOGRAPHY

AL-BAALI, M. and R. FLETCHER [1983]. Variational methods for nonlinear least squares, Report NA/71, University of Dundee, Department of Mathematical Sciences.

AL-BAALI, M. and R. FLETCHER [1985]. Variational methods for nonlinear least squares, *J. Opl. Res. Sos.*, 36, pp. 405-421.

APOSTOL, T. M. [1957]. *Mathematical Analysis,* Addison-Wesley, Reading, Massachusetts.

BETTS, J. T. [1976]. Solving the nonlinear least squares problem: Application of a general method. *J. Optim. Theory Appl.*, 18, pp. 469-484.

BOENDER, C.G.E., A.H.G. RINNOOY KAN, G.T. TIMMER, and L. STOUGIE [1982]. A stochastic method for global optimization, *Math. Programming,* 22, pp. 125-140.

BRODLIE, K.W. [1977]. Unconstrained minimization, in *The State of the Art in Numerical Analysis,* D.Jacobs, ed., Academic Press, London, pp. 229-268.

BROYDEN, C.G. [1965]. A class of methods for solving nonlinear simultaneous equations, *Math. Comp.*, 19, pp. 577-593.

BROYDEN, C.G. [1970]. The convergence of a class of double-rank minimization algorithms, Parts I & II, *IMA J. Appl. Math.* 6, pp. 76-90; 222-236.

BROYDEN, C.G. [1971]. The convergence of an algorithm for solving sparse nonlinear systems, *Math. Comp.*, 25, pp. 285-294.

BROYDEN, C.G., J.E. DENNIS Jr., and J.J. MORÉ [1973]. On the local and superlinear convergence of quasi-Newton methods *IMA J. Appl. Math.*, 12, pp. 223-246.

BUCKLEY, A.G. [1978]. A combined conjugate gradient quasi-Newton minimization algorithm, *Math. Programming,* 15, pp. 200-210.

BUCKLEY, A.G. [1978]. Extending the relationship between the conjugate gradient and BFGS algorithms, *Math. Programming,* 15, pp. 343-348.

CELIS, M.R., J.E. DENNIS Jr. and R.A. TAPIA [1984]. A trust region strategy for equality constrained optimization, TR84-1, Department of Mathematical Sciences, Rice University, Houston, Texas.

CHANKONG, V. and Y.Y. HAIMES [1983]. *Multiobjective Decision Making: Theory and Methodology,* North-Holland, New York.

COLEMAN, T.F. and J.J. MORÉ [1982]. Software for estimating sparse Jacobian matrices, Cornell Computer Science TR 82-502.

COLEMAN, T.F. and J.J. MORÉ [1983]. Estimation of sparse Jacobian matrices and graph coloring problems, *SIAM J. Numer. Anal.*, 20, pp. 187-209.

CONTE, S. and C. DE BOOR [1980]. *Elementary Numerical Analysis,* McGraw-Hill, New York.

CURTIS, A., M.J.D. POWELL and J.K. REID [1974]. On the estimation of sparse Jacobian matrices, *IMA J. Appl. Math.*, 13, pp. 117-120.

DAVIDON, W.C. [1959]. Variable metric methods for minimization, Argonne National Labs Report ANL-5990.

DAVIDON, W.C. [1980]. Conic approximations and collinear scalings for optimizers, *SIAM J. Numer. Anal.*, 17-2, pp. 268-281.

DEMBO, R.S., S.C. EISENSTAT and T. STEIHAUG [1982]. Inexact Newton methods, *SIAM J. Numer. Anal.*, 19, pp. 400-408.

DENNIS, J.E. Jr. [1973]. Some computational techniques for the nonlinear least squares problem, in *Numerical Solutions of Systems of Nonlinear Equations,* G.D. Byrne and C.A. Hall eds., Academic Press, New York, 1983, pp. 157-183.

DENNIS, J.E. Jr. [1977]. Nonlinear least squares and equations, in *The State of the Art in Numerical Analysis,* D. Jacobs, ed., Academic Press, London, pp. 269-312.

DENNIS, J.E. Jr. [1978]. A brief introduction to quasi-Newton methods, in *Numerical Analysis,* G.H. Golub and J. Oliger, eds., AMS, Providence, Rhode Island, pp. 19-52.

DENNIS, J.E. Jr., D.M. GAY and R.E. WELSCH [1981a]. An adaptive nonlinear least-squares algorithm, *TOMS,* 7, pp. 348-368.

DENNIS, J.E. Jr., D.M. GAY and R.E. WELSCH [1981b]. Algorithm 573 NL2SOL - An adaptive nonlinear least-squares algorithm [E4], *TOMS,* 7, pp. 369-383.

DENNIS, J.E. Jr. and H.H. MEI [1979]. Two new unconstrained optimization algorithms which use function and gradient values, *J. Optim. Theory Appl.*, 28, pp. 453-482.

DENNIS, J.E. Jr. and J.J. MORÉ [1974]. A characterization of superlinear convergence and its application to quasi-Newton methods, *Math. Comp.*, 28, pp. 549-560.

DENNIS, J.E. Jr. and J.J. MORÉ [1977]. Quasi-Newton methods, motivation and theory, *SIAM Review,* 19, pp. 46-89.

DENNIS, J.E., S. SHENG, and P.H. VU [1985]. A memoryless augmented Gauss-Newton method for nonlinear least-squares problems, TR85-1, Department of Mathematical Sciences, Rice University, Houston, Texas.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1979]. Least change secant updates for quasi-Newton methods, *SIAM Review,* 21, pp. 443-459.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1981]. A new derivation of symmetric positive definite secant updates, *Nonlinear Programming* 4, O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewood Cliffs, New Jersey.

DENNIS, J.E. Jr. and H.F. WALKER [1981]. Convergence theorems for least-change secant update methods, *SIAM J. Numer. Anal.,* 18, pp. 949-987.

FLETCHER, R. and M.J.D. POWELL [1963]. A rapidly convergent descent method for minimization, *Comput. J.,* 6, pp. 163-168.

FLETCHER, R. and C. XU [1985]. Hybrid methods for nonlinear least squares, Report NA/92, December 1985, Dept. of Mathematical Sciences, University of Dundee.

GAY, D.M. [1981]. Computing optimal locally constrained steps, *SIAM J. Sci. Statist. Comput.,* 2, pp. 186-197.

GAY, D.M. [1983]. A trust region approach to linearly constrained optimization, *Proceedings,* Dundee Biennial Conference on Numerical Analysis.

GAY, D.M. and R.B. SCHNABEL [1978]. Solving systems of nonlinear equations by Broyden's method with projected updates, in *Nonlinear Programming,* 3, O.Mangasarian, R.Meyer, and S. Robinson, eds., Academic Press, New York, pp. 245-281.

GILL, P.E., G.H. GOLUB, W. MURRAY and M.A. SAUNDERS [1974]. Methods for modifying matrix factorizations, *Math. Comp.,* 28, pp. 505-535.

GOLDFELDT, S.M., R.E. QUANDT, and H.F. TROTTER [1966]. Maximization by quadratic hill-climbing, *Econometrica,* 34, pp. 541-551.

GOLUB, G.H. and C.F. VAN LOAN [1983]. *Matrix Computation,* Johns Hopkins University Press, Baltimore, Maryland.

HALD, J. and K. MADSEN [1985]. Combined LP and quasi-Newton methods for nonlinear l_1 optimization, *SIAM J. Numer. Anal.,* 22, pp. 68-80.

HANSON, R.J. and F.T. KROGH [1983]. New algorithm for nonlinear least squares problems: part 1, Sandia Report SAND83-0936.

HEBDEN, M.D. [1973]. An algorithm for minimization using exact second derivatives, Technical report TP515, Atomic Energy Research Establishment, Harwell, England.

IGNIZIO, J.P. [1982]. *Linear Programming in Single- and Multiple- Objective Systems,* Prentice-Hall, Englewood Cliffs, N.J.

LEVENBERG, K. [1944]. A method for the solution of certain problems in least squares, *Quart. Appl. Math.,* 2, pp. 164-168.

LUENBERGER, D.G. [1973]. *Introduction to Linear and Nonlinear Programming,* Addison-Wesley, Reading, Massachusetts.

LYNESS, J.N. [1981]. Remarks about performance profiles, Technical Memorandum No. 369, Applied Mathematics Division, Argonne National Laboratory.

MANGASARIAN, O.L. [1969]. *Nonlinear Programming,* McGraw-Hill, New York.

MARQUARDT, D. [1963]. An algorithm for least-squares estimation of nonlinear parameters, *SIAM J. Appl. Math.,* 11, pp. 431-441.

MARWIL, E.S. [1978]. Exploiting sparsity in Newton-type methods, Cornell Applied Math. Ph.D. Thesis.

MORÉ, J.J. [1977]. The Levenberg-Marquardt algorithm: implementation and theory, *Numerical Analysis,* G.A. Watson, ed., pp. 105-116, Lecture Notes in Mathematics 630, Springer-Verlag, Berlin, Heidelberg and New York.

MORÉ, J.J. [1982]. Recent developments in algorithm and software for trust region methods, Argonne National Labs Report ANL/MCS-TM-2.

MORÉ, J.J., B.S. GARBOW and K.E. HILLSTROM [1980]. User guide for MINPACK-1, Argonne National Labs Report ANL-80-74. Available from National Technical Information Service, Springfield, Virginia.

MORÉ, J.J. and D.C. SORENSEN [1979]. On the use of directions of negative curvature in a modified Newton method, *Math. Prog.*, 16, pp. 1-20.

MORÉ, J.J. and D.C. SORENSEN [1982]. Newton's method, Argonne National Labs Report ANL-82-8, Argonne, Illinois.

MORÉ, J.J. and D.C. SORENSEN [1983]. Computing a trust region step, *SIAM J. Sci. Statist. Comput.*, 4, pp. 553-572.

NAZARETH, L. [1980]. Some recent approaches to solving large residual nonlinear least squares problems, *SIAM Rev.*, 22, pp. 1-11.

NAZARETH, L. [1983]. An adaptive method of minimizing a sum of squares of nonlinear functions [1983]. WP-83-99 International Institute for Applied Systems Analysis.

OREN, S.S. [1973]. Self-scaling variable metric algorithms without line search for unconstrained minimization, *Math Comput*, 27, 873-885.

OREN, S.S. [1974]. On the selection of parameters in self-scaling variable metric algorithms, *Math. Programming*, 7, pp. 351-367.

OREN, S.S. and D.G. LUENBERGER [1974]. Self-scaling variable metric (SSVM) algorithms. I. Criteria and sufficient conditions for scaling a class of algorithms, *Manage. Sci.* 20, pp. 845-862.

ORETEGA, J.M. and W.C. RHEINBOLDT [1970]. *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York.

PARKS, T.A. [1985]. Reducible nonlinear programming problems, TR 85-8, May 1985, Rice University.

POWELL, M.J.D. [1970a]. A hybrid method for nonlinear equations, in *Numerical Methods for Nonlinear Algebraic Equations,* P. Rabinowitz, ed., Gordon and Breach, London, pp. 87-114.

POWELL, M.J.D. [1970b]. A new algorithm for unconstrained optimization, in *Nonlinear Programming,* J.B. Rosen, O.L. Mangasarian and K. Ritter, eds., Academic Press, New York, pp. 31-65.

POWELL, M.J.D. [1970c]. A Fortran subroutine for solving systems of nonlinear algebraic equations, *Numerical Methods for Nonlinear Algebraic Equations,* P. Rabinowitz, ed., Gordon and Breach, London, pp. 115-149.

POWELL, M.J.D. [1975]. Convergence properties of a class of minimization algorithms, in *Nonlinear Programming 2* O.L. Mangasarian, R.R. Meyer, S.M. Robinson, eds., Academic Press, New York, pp. 1-27.

POWELL, M.J.D. [1976]. Some global convergence properties of a variable metric algorithm without exact line searches, in *Nonlinear Programming,* R. Cottle and C. Lemke, eds., AMS, Providence, Rhode Island, pp. 53-72.

POWELL, M.J.D. [1981]. An upper triangular matrix method for quadratic programming, *Nonlinear Programming 4,* Academic Press, New York, pp. 1-24.

POWELL, M.J.D. [1981]. A note on quasi-Newton formulae for sparse second derivative matrices, *Math. Programming,* 20, pp. 144-151.

POWELL, M.J.D. [1984]. On the global convergence of trust region algorithms for unconstrained minimization, *Math. Programming,* 29, pp. 297-303.

POWELL, M.J.D. and PH.L. TOINT [1979]. On the estimation of sparse Hessian matrices, *SIAM J. Numer. Anal.,* 16, pp. 1060-1074.

REID, J.K. [1973]. Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm, in *Proceedings* of the NATO Conf. at Cambridge, July 1972, North Holland, Amsterdam, pp. 437-445.

SCHNABEL, R.B. and P.D. FRANK [1984]. Tensor methods for nonlinear equations, *SIAM J. Numer. Anal.,* 21-5, pp. 815-843.

SCHULTZ, G.A., R.B. SCHNABEL and R.H. BYRD [1985]. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties, *SIAM J. Numer. Anal.,* 22, pp. 47-67.

SORENSEN, D. [1980]. The Q-superlinear convergence of a collinear scaling algorithm for unconstrained optimization, *SIAM J. Numer. Anal.,* 17-1, pp. 84-114.

SORENSEN, D.C. [1982a]. Trust region methods for unconstrained minimization, in *Nonlinear Optimization* 1981, M.J.D. Powell, ed., Academic Press, London.

SORENSEN, D.C. [1982b]. Newton's method with a model trust region modification, *SIAM J. Numer. Anal.,* 19, pp. 409-426.

STEIHAUG, T. [1980]. Quasi-Newton methods for large scale nonlinear problems, Ph.D dissertation, SOM Technical Report #49, Yale University.

STEIHAUG, T. [1981]. The conjugate gradient method and trust regions in large scale optimization, Department of Mathematical Sciences, TR81-1, Rice University.

STEWART, G.W. III [1973]. *Introduction to Matrix Computations,* Academic Press, New York.

VARDI, A. [1980]. *Trust region strategies for unconstrained and constrained minimization,* Ph.D. Thesis, School of Operations Research and Industrial Engineering, Cornell University.

WOODS, D. [1985] An interactive approach for solving multi-objective optimization problems, Department of Mathematical Sciences, TR85-5, Rice University.

APPENDIX A

GLOSSARY

The following is a brief glossary of terminology used in this thesis. Selected section references are also presented.

1. Abbreviations and Acronyms

BFGS	The most widely used secant method. See Section 1.1.1.
DFP	Another widely used secant method. See Section 1.1.1.
DGW [1981]	Dennis, Gay, Welsch [1981 a,b]. See Section 1.1.1.
FCD step	A step for which the model predicts a decrease of at least a fraction of the decrease predicted for the Cauchy point. See Section 4.2.1.
FOSPC	First order stationary point convergence. See Section 2.2.2.
NLS	The nonlinear least squares problem. See Section 1.1.1.
NL2SOL	The model switching algorithm presented in DGW [1981] for the solution of NLS problems. See Section 1.1.1.
OLC step	An optimal locally constrained step. See Sections 2.3.1, 4.2.1.
SSB [1985]	Schultz, Schnabel, Byrd [1985]. See Section 2.4.
TRS	The trust region subproblem. See Section 2.3.
UPD	Uniformly predicted decrease. See Section 5.2.
WFOSPC	Weak first order stationary point convergence. See Section 2.2.2.

2. Basic Notation

See also : Special symbols and parameters, Section 3 of this Appendix.

$ared_k(p)$ The actual function reduction $f(x_k)-f(x_k+p)$. See equations (2.1c), (4.10).

$A(i,\Delta_k)$ The set of "acceptable" steps at iteration k for model i with trust radius Δ_k. See Section 4.2.1.

$A(\Delta_k)$ The set of "acceptable" steps at iteration k with trust radius Δ_k, where the model is understood from context. See Section 4.2.1.

$B_k, B_k^i \in \mathbb{R}^{n\times n}$ An approximation to $H(x_k)$. Superscripts are used to distinguish between different model Hessians at the same iteration k. See equations (2.2b), (4.1).

Δ_k The trust radius at iteration k. See equation (TRS), Section 2.3.

$\bar{\Delta}_k$ The trial trust radius at iteration k. See Algorithm A.4.3.

$e_k(p), e_k^i(p)$ The observed error for a step p : $e_k^i(p) \equiv |\, ared_k(p) - pred_k^i(p)\,|$ Superscripts are used to distinguish between different models at the same iteration k. See equations (3.4), (5.2).

$f:\mathbb{R}^n \rightarrow \mathbb{R}^1$ The function to be optimized. See equation (1.1).

$g:\mathbb{R}^n \rightarrow \mathbb{R}^n$ The gradient of f.

$g_k \in \mathbb{R}^n$ Denotes $g(x_k)$.

$H:\mathbb{R}^n \rightarrow \mathbb{R}^{n\times n}$ The Hessian of f.

$H_k \in \mathbb{R}^{n\times n}$ Denotes $H(x_k)$.

$\bar{i}$ The initial trial model at a given iteration. See Algorithm A.4.3.

I_k A set of indices of models under consideration at iteration k. See equation (5.3).

$\hat{I}_k$ A set of indices of models at iteration k. Typically, $\hat{I}_k$ is the set of models for which steps have been computed. See equations (5.4), (5.5).

$\tilde{I}_k$ A set of indices of models at iteration k. Typically, $\hat{I}_k$ is the set of models for which trial steps have *not* been computed. See equations (5.4), (5.5).

$J:\mathbb{R}^n \rightarrow \mathbb{R}^{n\times m}$ The Jacobian of r.

$J_k \in \mathbb{R}^{n\times m}$ Denotes $J(x_k)$.

$L(f,x_1)$ The level set of f at x_1. See Definition 2.6.

$\hat{L}(f,x_1)$ The convex hull of the level set of f at x_1. See Definition 2.6.

$\tilde{L}(f,x_1)$ An open convex set containing $\hat{L}(f,x_1)$. See Section 4.2.2, or Section 4 of this Appendix.

N_k The number of models in the index set I_k. See equations (4.1), (5.3).

Ω_m The intersection between $L(f, x_1)$ and a certain ball about x_m. See equation (4.19).

$p, p_k, p_k^i \in \mathbb{R}^n$ A step away from x_k. The superscript denotes that p_k^i is associated with the model having index i.

$p \in A(i, \Delta_k)$ An ``acceptable'' step at iteration k for model i with trust radius Δ_k. See Section 4.2.1.

$p \in A(\Delta_k)$ An ``acceptable'' step at iteration k with trust radius Δ_k, where the model is understood from context. See Section 4.2.1.

$\phi_k^i(x)$ A model of f with index i at iteration k. See equations (2.1a), (2.2a).

$pred_k(p), pred_k^i(p)$ Predicted function reduction : $pred_k^i(p) = f(x_k) - \phi_k^i(x_k + p)$. See equation (2.1b). For standard quadratics, this becomes : $pred_k^i(p) = -g_k^i p - \frac{1}{2} p^t B_k^i p$. See equations (2.1b), (4.1). Superscripts are used to distinguish between different models at the same iteration k.

$\rho_k(p), \rho_k^i(p)$ Ratio of actual reduction to predicted reduction : $\rho_k^i(p) \equiv \dfrac{ared_k(p)}{pred_k^i(p)}$. Superscripts are used to distinguish between different models at the same iteration k. See equations (3.2), (4.11).

$r : \mathbb{R}^n \to \mathbb{R}^m$ The residual function of the nonlinear least squares problem. See equation (1.2).

$r_k \in \mathbb{R}^m$ Denotes $r(x_k)$.

$S : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ The term in the Hessian of the nonlinear least squares problem involving second order derivatives of the components of r. See equation (1.5).

τ The trust radius increase or decrease factor. At the end of an iteration, $\bar{\Delta}_{k+1}$ is set to either $\tau \Delta_k$ or $\tau \| p_k \|$ depending on the algorithm used. See Algorithms AS.4.6, AS.5.5.

$x \in \mathbb{R}^n$ A vector in $\mathbb{R}^n$.

$x^* \in \mathbb{R}^n$ Depending on context, a local minimizer of f or a first order stationary point.

$x_k \in \mathbb{R}^n$ The iterate at step k.

ξ_k An overestimate for the maximum curvature of the function f. See equations (6.1), (6.5), (6.46), (6.47) and Procedure A.6.2.

y_k A vector used with secant methods. Typically, $y_k = g_{k+1} - g_k$. See equations (6.3), (6.7) and (6.9).

ζ_k A sequence of numbers associated with the inexact quasi-Newton condition. They satisfy $0 \le \zeta_k < 1$ and $\lim_{k \to \infty} \zeta_k = 0$. See equation (4.6b).

3. Special Symbols and Parameters[1]

3.1. Parameters Relating to Trust Regions [2]

η_1 Minimum acceptable ratio of predicted reduction to actual reduction.

η_2 If the ratio of predicted reduction to actual reduction exceeds η_2, no trust radius decrease is allowed.

η_3 Ratio of predicted reduction to actual reduction used to trigger an increase in trust radius.

γ_0, γ_1 Limiters for the trust radius reduction factor τ. A typical algorithm will select $\tau \in [\gamma_0, \gamma_1]$ when a trust radius reduction is called for. However, most of our algorithms allow $\tau \in (0, \gamma_1]$.

γ_2, γ_3 Limiters for the trust radius increase factor τ. A typical algorithm will select $\tau \in [\gamma_2, \gamma_3]$ when a trust radius increase is called for.

σ_1 This constant determines the maximum step length as a function of the trust radius. All trust region algorithms enforce $\| p_k \| \le (1+\sigma_1)\Delta_k$.

[1] A parameter is a number used to define or limit an algorithm or procedure. For example, if an iterative procedure is considered to have failed after performing $k_{\max}$ iterations without convergence, the number $k_{\max}$ is said to be a parameter. The precise value of a parameter is typically defined only when an algorithm is implemented.

[2] The descriptions given here are very general and are intended only to aid readers unfamiliar with trust region methods. The precise use of these parameters sometimes changes for different algorithms, so the specifications and descriptions in the text of this thesis should be considered definitive.

TABLE 2

PARAMETERS FOR TRUST REGION METHODS

Symbol	Typical Values	Typical Legal Values	Sample Section References
η_1	0.001	$(0,1)$	2.4, 4.5.1, 4.5.2, 4.5.3
η_2	0.1	$(\eta_1,1)$	2.4, 4.5.2, 4.5.3
η_3	0.5	$(\eta_2,1)$	2.4, 4.5.1
γ_0	0.1	$(0,1)$	2.4
γ_1	0.5	$[\gamma_0,1)$	2.4 ,4.5.3
γ_2	2	$[1,\infty)$	2.4 ,4.8
γ_3	4	$[\gamma_2,\infty)$	2.4 ,4.6
σ_1	0.1	$(0,1)$	2.3.1, 4.2.1

3.2. Parameters Relating to Model Switching

μ A parameter used to control the relative frequency of model switching. NL2SOL uses $\mu=1.5$. See Sections 5.5, 5.7 and Algorithm AS.5.3.

η_1,η_2 These parameters are primarily "trust region parameters," but can also play a role in model switching. Typically, if the ratio of actual reduction to predicted reduction is less than η_2 the algorithm will consider switching models. If the ratio is less than η_1, the algorithm *must* either switch models or reduce the trust radius. See Section 5.5, Algorithm AS.5.3.

3.3. Constants[3] Associated with the Curvature of the Problem

β_1 Under the standard assumptions, we assume an upper bound on $\|H(x)\|$, say $\|H(x)\| \leq \beta_1$. See Section 4.2.2.

β_2 At every iteration, one model is typically assumed to be a standard quadratic with $\|B_k\| < \beta_2$. See Section 4.2.1.

β_3 The constant typically used with the UPD condition. See Sections 5.2 and 6.3.

β_4 A constant typically used with the UPD condition and secant methods. See Section 6.3.

ε_4 A constant typically associated with β_4. See Sections 6.3.2 and 6.3.4.

3.4. Parameters Relating to the Solution of the Trust Region Subproblem

c_1 Parameter associated with an FCD step. See equations (2.14), (4.2).

c_2 Parameter associated with an OLC step. See equations (2.12), (4.3).

c_3 Parameter associated with a UPD step. See equations (4.4), (5.1).

c_0 Parameter used to *enforce* the UPD condition. See equation (6.1).

3.5. Miscellaneous Parameters

j_c Integer parameter used in A.5.4 and A.6.1 to assure finite termination of the internal reduction loop. See Section 5.5.3.

$\bar{\delta}_0$ Parameter used in AS.5.6.

ε_0 Parameter used in *UPD_Test_OLC* for enforcing the UPD condition. See equation (6.47).

[3] A number is said to be a constant if it depends only on a specific set of parameters and a specific problem.

4. Standard Assumptions on the Function

For given $x_1 \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$, the following conditions are referred to as the standard assumptions on the function :

(SA1) $f(x)$ is twice continuously differentiable on an open convex set containing $L(f, x_1)$. This set is denoted $\tilde{L}(f, x_1)$.

(SA2) $f(x)$ is bounded below on $L(f, x_1)$.

(SA3) $\exists\ \beta_1 > 0$ such that for all $x \in \tilde{L}(f, x_1)$, the Hessian of f satisfies $\| H(x) \| \leq \beta_1$.

APPENDIX B

Of Things Not Treated

1. Further Useful Concepts

We now present several concepts of small importance to the theory of multi-model algorithms, but which are of use in actually writing an implementation.

A distinction can be made between algorithms which treat all models more or less equally, and those which know from the start which one is ''best'' in some context. For example, consider an algorithm using a constant matrix for the first model, and a sequence of matrices for the second. Suppose $\{\|B_k^2\|\}$ is not known to be bounded. The algorithm can be designed to treat them equally, or to ''fall back on'' the constant model under certain conditions. Logic which distinguishes between models based on a priori information is called **preferential** logic and logic which makes no such distinctions **nonpreferential** logic.

The program NL2SOL described in DGW [1981] uses nonpreferential logic for several reasons, among them the fact that it is generally not known in advance whether a NLS problem is small residual or large. Al-Baali, Fletcher [1985] present algorithms which treat the Gauss-Newton and secant models in a fundamentally different way. For example, some of their algorithms will permanently switch to the use of the secant model after a given number of iterations.

Another way to initially select a model is to use past performance as a guide. **Hereditary switching** is a switching system which gives preference to different models based on the results of tests that occurred in the previous iteration. **Nonhereditary switching** is a switching system which has no memory of performance of each model at the last iteration.

In NL2SOL the model used to test the step in the most recent iteration is tried first and given preference unless the final e - test in the most recent iteration rated the alternate as *significantly* better. NL2SOL uses this hereditary system with the justification that if a model is performing satisfactorily, one should stick to it unless there is good reason not to, particularly since such things as previous trust region updates have been determined via this model. Original test results as reported in DGW [1981] support this philosophy.

The following scheme for classifying models should clarify the different ways model generation is treated. The models are organized by how they relate to the multi-model algorithm which generates them.

Independent model generation is a system where models are generated or updated totally independently. In nonlinear least squares, such a system might take $J_k{}'J_k$ as one model and a BFGS secant approximation as an alternate.

Semi-independent model generation is a system where models may depend on others computed at this iteration, but not on any of the switching logic used in the last iteration. Some examples are as follows.

(a) In NLS problems, such a system might take $B_k = J_k{}'J_k$ as one model with alternate $B_k = J_k{}'J_k + S_k$. Here S_k is one of the structured secant approximations to $S(x)$.

(b) In NLS problems, another example of such a system is a BFGS algorithm which resets to $J_k{}'J_k$ after a *fixed* number of iterations.

Dependent model generation is a system where models may depend on others computed at this iteration, and on switching logic used in the previous iteration. This allows such things as "restarts." Some examples of this type of system are as follows.

(a) Al-Baali, Fletcher [1985] algorithms which reset S_k to zero whenever the Gauss-Newton Hessian is selected.

(b) The algorithm of Nazareth [1980,1983] which chooses a parameter α to interpolate between a secant update and the Gauss-Newton model based on performance of each model at the last step.

(c) MINPACK (Moré, Garbow, Hillstrom [1980]) routines which "refresh" a secant approximation by setting it to $H(x_k)$ when it appears to be behaving poorly.

2. General Definition for Models

More generality can be obtained by defining a model in terms of the processes which can be associated with it. Such a definition allows us to conceptually separate each different way a model can be used.

When referred to in context as an *abstract concept*, a *model* having indices i and k is something having one or more of the following processes associated with it.

(a) Evaluating for some given x the single valued function $\phi_k^i(x) : \Omega_k^i \rightarrow R$, with Ω_k^i a nonempty subset of $\mathbb{R}^n$, or

(b) Providing a trial step $p_k^i \in \mathbb{R}^n$, or

(c) Setting up information necessary for the execution of a process associated with a model having indices j and $k+1$.

Considering a model defined in this fashion delineates between "testing a trial step" as in (a), "computing a trial step" as in (b), and "updating" as in (c). Thus a model could be defined which only evaluates trial steps generated by other models without having the capability of generating such trial steps, or vice versa. Also, a model need not be called upon to do either at any given step, as long as it stores enough information to generate itself when needed.[1]

Commonly, any p_k^i provided by a model is associated with a function $\phi_k^i(x)$ which can be evaluated for various x, and $\phi_k^i(x_k+p_k^i)$ is automatically provided whenever p_k^i is provided. However, we will not rule out models that can only invoke processes to evaluate ϕ or only to provide a step p_k^i, or only to define a new model for the next step.

[1] For example, with the Newton model we need not actually evaluate the exact Hessian at each iteration unless it is actually needed. A more complex example is a model which at each iteration "remembers" the last m iterates and "updates" the identity matrix to produce a quadratic which interpolates this data as well as possible. Our "model" would be the background process of "remembering" roughly m vectors until needed.